Universidad de las Ciencias Informáticas

Centro de Informática Industrial , Facultad 5

Cliente Escritorio para el Entorno de Visualización de la Interfaz Hombre Máquina del SCADA SAINUX

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Alik Ramón del Risco del Risco

Tutores: Ing. Luis Andrés Valido Fajardo

Ing. Ridel Oscar García Mora

La Habana, 2016

Piensa diferente y serás diferente
Steve Jobs

Dedicatoria

A mis padres que son los mejores y no se como expresar toda la gratitud, amor, respeto y admiración que tengo hacia ellos, hoy soy la persona que soy por ellos.

A mi hermanita linda que la quiero con la vida que siempre me sirvió de ejemplo y le debo mucho a ella.

Finalmente a mi sobrino que es mucho mas que el parentezco familiar, porque espero muchas cosas de él, espero que le yo le pueda servir de ejemplo y algún dia nos regale a la familia un título universitario.

Agradecimientos

Le agradezco a toda mi familia, resaltando a mis padres, mi hermana, mi sobrino, a Mayda.

A mis tutores que los admiro mucho como profesionales ojala pueda seguir aprendiendo con ellos, especialmente a valido .

A todos mis amigos, que los voy a extrañar un mundo, sobre todo a los que vienen de primer año conmigo aguantandome Carlos, Oscar, Yalena, Brian, Lidia, Alexis, Elaine, Alberto, Leduan, Koky, Richard, Raul y para resumir que sino más nunca se termina, los del grupo, del apartamento, del edificio, del movimiento de programción competitiva (Nelson y el Pepe), las personas del centro, del fútbol (Yalbert que me enseñó a mejorar un poquito) , del MW3, del Dota, del WoW, del FIFA, a todos en el que en algun momento molesté (y mira que molesté) , y no voy a decir mas nombre para que nadie se ponga celoso al yo cometer el error de no mencionarlo.

A todos los profesores desde el círculo infantil hasta la universidad, mi profe preferida mi mamá, a la profe Yirka y Zaida.

Y muchos más que tendría que decir pero de verdad me pesa escribir y tener que leerlo y ya seguro en este momento estaría llorando y muchos riéndose, y quiero celebrarlo con todos los presentes y otros que no puedan estar aquí.

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Alik Ramón del Risco del Risco
Autor

Ing. Luis Andrés Valido Fajardo
Tutor

Ing. Ridel Oscar García Mora
Tutor

Resumen

En la Universidad de las Ciencias Informáticas se encuentra el Centro de Informática Industrial, que desarrolla el Sistema de Automatización Industrial basado en GNU/LINUX y está compuesto por varios módulos, entre ellos se destaca la Interfaz Hombre-Máquina que se divide por el Entorno de Configuración y el Entorno de Visualización. La presente investigación surge a partir de la necesidad de visualizar y supervisar la información del Sistema de Automatización Industrial basado en GNU/LINUX, en las estaciones sin importar el sistema operativo, y evitando la instalación de dependencias y bibliotecas del Entorno de Configuración y de los subsistemas que componen el Sistema de Automatización Industrial basado en GNU/LINUX. El objetivo fundamental es desarrollar una aplicación de escritorio capaz de visualizar en tiempo real la información del Sistema de Automatización Industrial basado en GNU/LINUX. Para el desarrollo de la solución propuesta se realiza un análisis de las principales herramientas, tecnologías y metodologías que se utilizan para la construcción de softwares. El proceso estuvo guiado por el uso de las siguientes herramientas y tecnologías: Visual Paradigm como herramienta de Ingeniería de Software Asistida por Computadora, Lenguaje Unificado de Modelado como lenguaje de modelado, C++ como lenguaje de programación respaldado por el marco de trabajo QT, haciendo uso de QT Creator como Entorno de Desarrollo Integrado. Para validar que los resultados alcanzados se realizaron pruebas de Aceptación y de Rendimiento, obteniendo así un cliente de escritorio capaz de representar en tiempo real los datos provenientes del Sistema de Automatización Industrial basado en GNU/LINUX.

Palabras clave: Entorno de Visualización, Interfaz Hombre Máquina, SCADA.

Índice general

Índice de figuras

Índice de tablas

Introducción

Desde mediados del siglo pasado hasta hoy en día ha crecido drásticamente la automatización de las industrias. Debido a que los procesos industriales son complejos y se necesita controlarlos rigurosamente, logrando una notable eficiencia y resultados en cuanto a seguridad, dinero, tiempo y materiales. En la vanguardia de la automatización de las industrias se encuentran los sistemas de Control, Supervisión y Adquisición de Datos (SCADA, por sus siglas en inglés). En la Universidad de las Ciencias Informáticas (UCI) en la Facultad 5 se encuentra el Centro de Informática Industrial (CEDIN) el cual se especializa en soluciones de automatización de procesos industriales, destacando entre los productos el Sistema de Automatización Industrial basado en GNU/LINUX (SAINUX). Dicho sistema está compuesto por varios módulos que trabajan de manera conjunta para llevar a cabo las tareas de supervisión, control y adquisición de datos, entre los cuales se encuentra el HMI. Dicho módulo se encarga de representar, en un ordenador, los procesos que ocurren en el campo, muestra los componentes implicados, los sensores, las estaciones remotas, y el sistema de comunicación dándole al operador total control. Permite al operador estar en contacto directo con el sistema, realizar la supervisión y el control del proceso en general. El HMI constan de dos entornos: el Entorno de Configuración (EC), donde los mantenedores configuran la información específica del área que se desea supervisar y diseñan los despliegues, los cuales haciendo uso de los componentes gráficos permiten simular los procesos de campo; el Entorno de Visualización (EV) es donde el operador puede supervisar y controlar la configuración realizada en el EC, interactuando con los componentes gráficos para emitir control sobre el sistema, monitorear los cambios de estado de las variables, gestionar alarmas, generar reportes. El SAINUX está orientado al trabajo en estaciones como una aplicación de escritorio, el cual debe ser instalado sobre el sistema operativo Debian 7 en cada ordenador destinado a las siguientes funciones la supervisión y control, o la configuración de los procesos. Esto trae consigo los siguientes inconvenientes :

- La solución está atada o depende del sistema operativo.
- En las estaciones de trabajo que están destinadas solamente a la supervisión y control se instala el entorno de configuración así como las dependencias y bibliotecas utilizadas por este.
- En las estaciones de trabajo se instalan las dependencias de los subsistemas del SAINUX, producto a que tanto el entorno de visualización y de configuración están fuertemente atados a estos.
- Producto al gran cúmulo de las dependencias que se instalan debido a lo explicado anteriormente se hace necesario que la estación de trabajo cuente con un alto nivel de prestaciones de hardware.

Por lo planteado anteriormente se propone el siguiente **problema de la investigación:** ¿Cómo elevar la

portabilidad[1] del Entorno de Visualización del SCADA SAINUX como aplicación de escritorio[2]?

Teniendo como **objeto de estudio:** La portabilidad del entorno de visualización de los sistemas SCADA.

Enmarcado en el **campo de acción:** La portabilidad del entorno de visualización de la Interfaz Hombre-Máquina en el SCADA SAINUX.

Se propone como **objetivo general:** Desarrollar un cliente de escritorio del entorno de visualización que permita elevar la portabilidad de la Interfaz Hombre-Máquina del SCADA SAINUX.

Para dar cumplimiento a los objetivos de esta investigación se definieron las siguientes **tareas investigativas**:

1. Establecimiento de los fundamentos teórico-metodológicos para el desarrollo de sistemas informáticos SCADA como entorno de escritorio.
2. Análisis del estado del arte de las principales tecnologías, metodologías y herramientas para el desarrollo de aplicaciones de escritorio en tiempo real.
3. Análisis y diseño de la solución informática.
4. Implementación de la solución informática.
5. Validación de los resultados obtenidos mediante la realización de pruebas.
6. Valoración cualitativa de los resultados obtenidos en la validación de la solución propuesta.

Durante la investigación se llevan a cabo varios métodos y técnicas en la búsqueda y procesamiento de la información como son:

Métodos Teóricos.

- **Analítico-sintético:** Para el estudio de los conceptos empleados en los sistemas SCADA, analizando todos los documentos elaborados, para la extracción de los elementos más importantes.
- **Histórico-lógico:** Para la comprensión de los antecedentes y las tendencias actuales referidas la evolución en el mundo de los sistemas SCADA.
- **Modelación:** Para la elaboración de múltiples diagramas para un mejor entendimiento del problema y solución.

Métodos Empíricos.

- **Observación:** Se puso en práctica este método para conocer el funcionamiento existente en los despliegues del SAINUX mediante el comportamiento de los dispositivos de campo en las propiedades de los objetos gráficos y sumarios empleados para la toma de decisiones de los operadores.
- **Entrevista:** Empleada para consultar con los especialistas del centro sobre algunos conceptos asociados con el negocio.

[1]Se define como la característica que posee un software para ejecutarse en diferentes plataformas, el código fuente del software es capaz de reutilizarse en vez de crearse un nuevo código cuando el software pasa de una plataforma a otra . A mayor portabilidad menor es la dependencia del software con respecto a la plataforma.

[2]Es una aplicación que se ejecuta sobre una computadora de escritorio o portátil.

- **Consulta bibliográfica:** Empleada para consultar las fuentes de información relacionados con los tipos de los sumarios y objetos gráficos visualizados en los entornos de escritorio de los HMI en sistemas SCADA.

CAPÍTULO 1

Fundamentación Teórica

Introducción

En el presente capítulo se engloban conceptos fundamentales asociados a los sistemas SCADA. Además se exponen las características principales de estos sistemas, específicamente en el entorno de visualización del módulo HMI. Se analizan las principales tendencias, tecnologías, metodologías y software utilizados en la actualidad para el desarrollo de aplicaciones de este tipo. A su vez se analizan y se fundamenta la selección de estas para el desarrollo de aplicaciones de la solución propuesta.

1.1. Sistemas de Supervisión, Control y Adquisición de Datos.

Un SCADA se define como una aplicación de software para ordenadores que permite controlar y supervisar procesos a distancia. Proporcionando comunicación con los dispositivos de campo y controlando el proceso de forma automática desde la pantalla del operador. Además, provee a diversos usuarios, toda la información que se genera en el proceso productivo, control de calidad, supervisión, mantenimiento. Son sistemas diseñados para ser implementados o usados en numerosas industrias, contando con módulos de software genéricos para proporcionar las capacidades requeridas por las distintas entidades. La mayoría de los SCADA que se instalan en la actualidad, se han convertido en una parte integral en la gestión de la información corporativa. Estos sistemas no son solamente herramientas operacionales, sino un recurso importante de información y esto se pone de manifiesto en los recursos que se exportan a otros sectores como la plataforma web y la tecnología móvil, permitiendo a usuarios externos ver lo que sucede en el proceso de producción controlado por el SCADA, así como obtener distintos reportes relacionados con datos importantes. Además de lo antes mencionado los actuales SCADA alcanzan un nivel aceptable de tolerancia a fallos y cuentan con ordenadores redundantes operando en paralelo con el mismo SCADA permitiendo la transferencia automática de la responsabilidad del control de cualquier ordenador que pueda tener cualquier tipo de colisión o falla por cualquier razón, a una computadora de reserva en línea permitiendo que no se detenga el servicio bajo ningún concepto. Los SCADA deben cumplir ciertos y determinados requerimientos para su fácil ins-

talación, configuración, mantenimiento y puesta en funcionamiento. Se puede decir que deben ser sistemas de arquitectura abierta, capaces de crecer o adaptarse según las necesidades cambiantes de la empresa. También deben comunicarse con toda facilidad y de forma transparente al usuario con el equipo de planta y con el resto de la empresa (redes locales y de gestión). Además deben ser programas sencillos de instalar, sin excesivas exigencias de hardware, y fáciles de utilizar, permitiendo interfaces de usuario amigables.

Entre sus principales funcionalidades se destacan:

- Adquisición de datos: Significa que el sistema tiene la capacidad de obtener información desde el sistema de control, por ejemplo, sobre valores de temperatura o presión y a diferencia de la supervisión los datos son registrados o almacenados para su posterior explotación.
- Supervisión: Significa poder observar o monitorear aquello que sucede en el proceso industrial, el equipo o la maquinaria, por ejemplo, conocer (generalmente de una forma gráfica) si un motor se encuentra encendido o apagado.
- Control: Significa tener la posibilidad de ejecutar comandos; en otras palabras poder enviar instrucciones hacia el sistema de control.

Los sistemas SCADA se componen por los siguientes módulos:

- Manejadores: Aseguran mediante una interfaz genérica la comunicación del sistema de supervisión y control con los distintos dispositivos que existen en el campo, ya sean autómatas, sensores inteligentes, etc.
- Núcleo de Procesamiento de Datos: Representa el núcleo principal del procesamiento de los datos, es el encargado del procesamiento y análisis de la información recogida del campo a través de los manejadores. Una vez procesada esta información, es enviada al módulo que la requiera.
- Base de Datos Históricos: Aquí se implementa el mecanismo encargado del almacenamiento de la información recibida desde el campo, así como la sucesión de alarmas y eventos generados. La información almacenada es utilizada por varias aplicaciones del sistema.
- Comunicación: Es la capa de software, que se encarga de la comunicación entre los diferentes procesos distribuidos de medio y alto nivel, que forman parte del sistema SCADA y el principal elemento que caracteriza su complejidad es la gestión de las comunicaciones.
- HMI: Se encarga de representar, en un ordenador, los procesos que ocurren en el campo, muestra los componentes implicados, los sensores, las estaciones remotas, y el sistema de comunicación dándole al operador total control. Este módulo es el que permite al operador estar en contacto directo con el sistema, realizar la supervisión y el control del proceso en general.(Penin, 2011)

1.1.1. Módulo HMI:

Se encarga de representar, en un ordenador, los procesos que ocurren en el campo, muestra los componentes implicados, los sensores, las estaciones remotas, y el sistema de comunicación dándole al operador

total control. Este módulo es el que permite al operador estar en contacto directo con el sistema,realizar la supervisión y el control del proceso en general.(Penin, 2011) Está compuesto por dos partes fundamentales:

EC: Permite configurar varios procesos o partes de ellos,aquí se definen y gestionan las variables, los manejadores, los comandos, las alarmas y variadas opciones adicionales. Este entorno funciona como una aplicación de diseño tradicional, con la peculiaridad que los sinópticos se confeccionan a partir de objetos y primitivas básicas predefinidas,que se pueden agrupar, combinar, transformar, importar y exportar entre otras.(ibíd.)

EV: Se encarga de visualizar las animaciones y los objetos definidos en el editor, muestra lo que está ocurriendo en el campo en tiempo real, es el que envía los comandos a las estaciones remotas, quién recibe los valores de las variables, interactúa con la mayoría de los operadores pues se emplea para supervisar el proceso de manera directa.(ibíd.)

1.2. Conceptos Asociados

Variable analógica: Una variable analógica es aquella que toma infinitos valores entre dos puntos cualesquiera de la misma. Un ejemplo de variable analógica es la temperatura. Si se analiza su variación durante un día, se observa que no puede pasar de un valor a otro dando un salto. Esto quiere decir que si la temperatura se incrementa de 11 grados a 17 grados, toma infinitos valores intermedios.(García, 2013)

Alarma: Es la capacidad de reconocer eventos excepcionales dentro del proceso y reportar estos eventos. Las alarmas son reportadas basadas en límites de control preestablecidos.(Martínez, 2013)

Variable digital: Una variable digital es aquella que solamente puede tomar un determinado número de valores. Un ejemplo de variable digital puede ser una alarma lanzada en un sistema informático para indicar alguna anomalía. Dentro de los tipos de variables digitales se encuentran las variables binarias las cuales toman solamente un estado de dos posibles. (García, 2013)

Comando: Señales que se envían a los dispositivos y que ejecutan un cambio de estado en una variable. Ofrecen la posibilidad de que los operadores puedan cambiar consignas u otros datos claves del proceso directamente desde el ordenador, puede ser marcha, paro, modificación de parámetros, entre otros. De esta forma se escriben datos sobre los elementos de control.

Acción operacional: Acción de modificar atributos de los recursos del sistema SCADA, desde interfaces de operador; por ejemplo modificar contraseñas, sacar de barrido un dispositivo, inhibir alarmas, entre otros. Estas acciones pueden ser realizadas estando el sistema en frío o en caliente.

Sumario: Representa en forma tabulada la información referente a las mediciones o datos que el sistema genera u obtiene ya sea desde los dispositivos de campo o de las bases de datos. Dicha mediciones o datos pueden agruparse o clasificarse en alarmas, eventos, estado del sistema, puntos, reportes, despliegues, estadísticas.

Despliegue: Es un resumen esquematizado con la ventaja de permitir visualizar la estructura y organización de los elementos que componen el proceso, así como el comportamiento de los mismos, los cuales

pueden estar ubicados en un área geográficamente extensa para que el operador monitoree constantemente cada de uno de los componentes que conforman el proceso.

1.3. Análisis sistemas existentes

1.3.1. Desarrollados en otros países:

InTouch es un SCADA de la rama Wonderware de la división de Administración de Operaciones de la compañía inglesa Invensys. Permite configurar alarmas y establecerles hasta 999 niveles de prioridad y hasta 8 niveles de jerarquía entre grupos de alarma con posibilidad de hasta 16 subgrupos para cada uno de ellos. Posibilita visualizar todas o un extracto de ellas de forma histórica y grabar en disco o imprimir en diferentes formatos personalizables. Las funciones de alarmas distribuidas incluyen reconocimiento global o selectivo, desplazamiento por la lista y visualización de alarmas procedentes de diferentes servidores en un único panel.(MeasureSoft., 2009)

ScadaPro de la compañía irlandesa Measurement Development Ltd. por su parte incluye un componente de Procesamiento y Gestión de alarmas cuyas características más importantes en el procesamiento de alarmas son:

- A cada canal se le pueden asignar condiciones de alarmas únicas para límites alto y bajo.
- Se evitan la ocurrencia de múltiples alarmas en un mismo canal empleando histéresis y retardo de alarmas.
- A cada condición de alarma se le puede asignar una prioridad en un rango de 1 a 255 y se le pueden asociar bloques de texto que son mostrados en caso de activarse la alarma.
- Varios canales de alarmas pueden estar vinculados a un canal de salida de alarmas común para propósitos de notificación, o para apagar partes importantes del proceso o la planta de forma automática.
- Las alarmas pueden ser reconocidas de forma independiente o en grupo.

(Pro, 2016)

Simantic WinCC es un sistema SCADA con funciones eficientes para el control de procesos automatizados. Funcionalidad operativa y el seguimiento completo bajo el sistema operativo Windows para todos los segmentos de la industria que van desde sistemas monopuesto simples hasta sistemas multiusuario distribuidos con servidores redundantes. Una de las características especiales es su total apertura. Se puede utilizar fácilmente en combinación con programas estándar y de usuario, la creación de soluciones HMI que cumplan con precisión los requisitos prácticos.(Siemens, 2016)

Ignition es un sistema SCADA liberado por Inductive Automation en 2010, basado en una arquitectura de base de datos SQL-céntrico. Cuenta con la implementación basada en cruz plataforma web a través de Java Web Start Technology. Está compuesto por tres componentes principales: puerta de enlace, diseñador y clientes en tiempo de ejecución. Sus módulos independientes proporcionan funcionalidad independiente en cualquier o todos los componentes de la plataforma, posee características tales como: control de estado en

tiempo real, alertas, informes, adquisición de datos, secuencias de comandos, programación y el apoyo móvil. Su módulo HMI es capaz de generar reportes de forma dinámica en pdf y permite el acceso a dispositivos móviles.(Automation, 2013)

RSView32 es un HMI integrado, basado en componentes para la supervisión y el control de máquinas de automatización y procesos. Es un sistema abierto que permite a los datos de la planta para ser compartidos con otros sistemas de fabricación, el suministro de información en toda la empresa de fabricación en tiempo real.

HMI500 La aplicación corresponde a la más reciente solución de EFACEC para la implementación de la Interfaz Hombre-Máquina de sistemas SCADA destinada con propósito para gestionar localmente, vía web, complejos sistemas distribuidos de automatización, supervisión, control y protección. La aplicación es del tipo web server y puede ser utilizada en diferentes tipos de plataforma de hardware, funcionando bajo el sistema operativo WINDOWS XP.(Silva, 2012)

1.3.2. Desarrollados en Cuba:

EROS es un Sistema de Supervisión y Control de Procesos Industriales de la Empresa de Servicios de Computación, Comunicaciones y Electrónica del Níquel Rafael Fausto Orejón Forment en Nicaro, Mayarí, provincia Holguín. Realiza variadas funciones dentro del entorno de la dirección de los procesos. Puede trabajar acoplado a diversos sistemas de colección de datos, como elemento único o formando parte de una red industrial. Tiene en cuenta todas las características de las variables medidas y permite realizar tratamiento estadístico y determinístico de las mismas. Permite la ejecución de comandos para el control manual y para realizar cambios de parámetros y acción de los reguladores. Su aplicación más importante es su uso en la vigilancia en tiempo real desde el Despacho Nacional de la Unión Eléctrica de la energía que consume cada uno de los 169 municipios del país.(SERCONI, 2010)

Guardián del Alba (Galba) es desarrollado en Venezuela por un equipo compuesto por personal tanto venezolano como cubano, en el cual toman parte estudiantes y profesores del CEDIN. Es un SCADA distribuido, comprendido por varios módulos. A partir de un análisis del sistema en ejecución y su código fuente se comprobaron sus funcionalidades. Realiza procesamiento de puntos recolectados y calculados, tanto analógicos como digitales. Detecta la presencia de condiciones anormales y las notifica al HMI. Soporta tratamiento para los siguientes tipos de alarmas: nivel, tasa de cambio, falla de instrumento, falla de comunicación, falla de no variación en tiempo y cambio de estado. El sistema procesa comandos de escritura de puntos y de manipulación de alarmas.(Jaime Fardales Pérez, 2008)

Telenul es desarrollado por la Empresa de Ingeniería y Proyectos de la Electricidad para el control de recerradores y seccionalizadores automáticos de industrias NULEC en la Unión Nacional Eléctrica. Permite telecontrolar y supervisar de forma remota los interruptores NULEC instalados en sus subestaciones y redes de distribución. El sistema corre sobre Windows y está programado en Delphi.(Durán, 2009)

SAINUX es un sistema distribuido, compuesto por diferentes subsistemas o módulos que trabajan de manera conjunta para llevar a cabo las tareas de supervisión, control y adquisición de datos. Cada uno de

estos módulos es encargado de realizar una tarea específica:

- **Comunicación:** Se encarga de la comunicación entre los diferentes módulos que forman parte del sistema. Implementa dos formas de comunicación, el modelo de comunicación asincrónica (Moore, 1994),el cual tiene como base el Servicio de Notificación de TAO, fundamentado en el paradigma publicación suscripción y el modelo sincrónico inherente al modelo cliente/servidor de la especificación utilizada (Gelbmann, 2002).
- **Configuración:** El módulo de configuración está formado por varios componentes o elementos de la configuración del proyecto relacionado con seguridad, comunicación, con los dispositivos de campo, variables entre otros. Es el encargado de almacenar, persistir y suministrar la información base para el funcionamiento de los demás módulos del SCADA.
- **Adquisición o núcleo de procesamiento de datos:** El núcleo de procesamiento de datos es el encargado del tratamiento y análisis en tiempo real de la información adquirida desde el campo.
- **Seguridad:** Este módulo proporciona las funcionalidades necesarias para garantizar el trabajo autorizado a usuarios y módulos. Tiene implementada herramientas que protegen al sistema de ataques maliciosos o involuntarios por parte de personas o recursos, tales como fallas de energía y problemas de red.
- **Histórico:** Se encarga del almacenamiento y procesado ordenado de los datos, de forma que otra aplicación o dispositivo pueda tener acceso a ellos, para realizar posteriores análisis como diagnósticos o reportes.
- **Interfaz Hombre Máquina:** Este módulo se encarga de representar en un ordenador, los procesos que ocurren en el campo en tiempo real. Muestra de forma gráfica los procesos operacionales, alarmas, variables, reportes y despliegues que permiten al operador el contacto directo con el sistema.

1.3.3. Capa de Servicios Web:

Es una aplicación desarrollada en php, alojada sobre un servidor Apache, que permite la conexión de varios clientes que se ejecutan en diversas plataformas y desarrollados en distintas tecnologías con el Servidor HMI.

1.3.4. Servidor HMI:

Software de HMI para brindar información y atender las solicitudes que se realizan desde clientes HMI y es desarrollado por el CEDIN de la UCI. Este sistema brinda un conjunto de funcionalidades para la supervisión y control de procesos industriales entre las que destacan: el envío a los clientes de la información referente a los sumarios de puntos, alarmas, subcanales y dispositivos. Permite atender las solicitudes de ejecución de acciones operacionales y de control sobre el sistema. Para su desarrollo se utiliza el lenguaje C++ y el marco de trabajo QT, está disponible para Debian Wheezy.

1.3.5. HMI QT

Software de HMI para la supervisión y control empleando el SCADA SAINUX. Está diseñado para ejecutarse en plataformas Linux y es desarrollado por el CEDIN de la UCI. Este sistema brinda un conjunto de funcionalidades para la supervisión y control de procesos industriales entre las que destacan: visualización de información en tiempo real en sumarios de puntos, alarmas, subcanales y dispositivos. Permite la ejecución de acciones operacionales y de control sobre el sistema a través de interfaces de detalles de puntos y dispositivos. Este sistema es capaz de mostrar despliegues o sinópticos y gráficos de tendencias. Requiere para su ejecución de requisitos mínimos un procesador Core i3 y 1 Gb de RAM. Para su desarrollo se utiliza el lenguaje C++ y el marco de trabajo QT, está disponible para Debian Wheezy.

1.3.6. HMI WEB

Software de HMI para la supervisión y control empleando el SCADA SAINUX. Está diseñado para ejecutarse en un navegador web sin necesidad de software adicional sobre cualquier plataforma y es desarrollado por el CEDIN de la UCI. Este sistema brinda un conjunto de funcionalidades para la supervisión y control de procesos industriales entre las que destacan: visualización de información en tiempo real en sumarios de puntos, alarmas, subcanales y dispositivos. Permite la ejecución de acciones operacionales y de control sobre el sistema a través de interfaces de detalles de puntos y dispositivos. Este sistema es capaz de mostrar despliegues o sinópticos y gráficos de tendencias. Requiere para su ejecución de requisitos mínimos navegador web Google Chrome versión 35.0 o superior, Core i3 y 512 Mb de RAM. Para su desarrollo se utiliza el lenguaje HTML, Javascript, y los marco de trabajo AngularJS y JQuery.

1.4. Tecnologías y herramientas para la implementación.

1.4.1. Proceso Unificado Ágil

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y de ayudas para la documentación en el desarrollo de productos software. En este se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben tener. Actualmente existen muchas de estas que están caracterizadas en distintas tendencias de desarrollo de software entre las cuales se encuentra Proceso Unificado Ágil (AUP, por sus siglas en inglés). AUP es una versión simplificada de Proceso Unificado Racional (RUP, por sus siglas en inglés). En él se describe un método sencillo, fácil de entender para el desarrollo de software de aplicaciones empresariales utilizando técnicas ágiles y conceptos que aún permanecen todavía fieles a RUP. Esta metodología abarca siete flujos de trabajo, cuatro de ellos son ingenieriles y los otros tres de apoyo: Modelado, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyectos y Ambiente. Dispone además de cuatro fases igual que RUP: Incepción o Creación, Elaboración, Construcción y Transición. Este enfoque incluye diversas técnicas ágiles, Desarrollo Dirigido por Pruebas,

Modelado Ágil, Gestión del Cambio Ágil y Refactorización de Base de Datos para mejorar su productividad. Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto en la UCI se decide hacer una variación de AUP para lograr que se adapte al ciclo de vida definido para la actividad productiva de la universidad y aumentar la calidad del software que se produce, debido a esto se propone utilizar **AUP-UCI** como metodología de desarrollo debido a que está definida por el CEDIN. De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega una fase de Cierre. **Objetivos de las fases (Variación AUP-UCI):**

- **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo, costo y decidir si se ejecuta o no el proyecto.
- **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.
- **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 8 disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación y la disciplina Despliegue se considera opcional. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto).

Objetivos de las Disciplinas (Variación AUP-UCI):

- **Modelado de negocio (opcional):** El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para garantías de que el software desarrollado va a cumplir su propósito.
- **Requisitos:** El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.

- **Análisis y diseño:** En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis.
- **Implementación:** En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.
- **Pruebas internas:** En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar las pruebas.
- **Pruebas de liberación:** Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
- **Pruebas de Aceptación:** Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.
- **Despliegue (Opcional):** Constituye la instalación, configuración, adecuación, puesta en marcha de soluciones informáticas y entrenamiento al personal del cliente.

AUP propone 9 roles (administrador de proyecto, ingeniero de procesos, desarrollador, administrador de la base de datos, modelador ágil, administrador de la configuración, stakeholder[1], administrador de pruebas, probador), se decide para el ciclo de vida de los proyectos de la UCI tener 11 roles, manteniendo algunos de los propuestos por AUP y unificando o agregando otros, teniendo como resultado los siguientes roles: jefe de proyecto, planificador, analista, arquitecto de información (opcional), desarrollador, administrador de la configuración, stakeholder, administrador de calidad, probador, arquitecto de software, administrador de la base de datos.(Sánchez, 2014)

1.4.2. Lenguaje Unificado de Modelado (UML):

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) en su versión 2.0 es un lenguaje estándar de modelado visual que se usa para especificar, construir, documentar y visualizar artefactos de un sistema de software. UML permite a los desarrolladores visualizar el resultado de su trabajo en esquemas o diagramas estandarizados. Por ejemplo, símbolos o iconos característicos utilizados para capturar los requisitos. Estos iconos no son más que una notación gráfica, es decir, una síntesis; sin embargo, detrás de esta notación gráfica UML especifica un significado, es decir, una semántica. Dicho lenguaje proporciona un vocabulario que incluye tres categorías: elementos, relaciones y diagramas, conteniendo aspectos conceptuales

[1]Es para definir a todos las personas interesadas en el proyecto, incluidos los usuarios, administración, personal de operaciones y personal de apoyo.

tales como procesos de negocios, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.(Booch; Rumbaugh; Jacobson; J. S. Martínez y Molina, 1999)

Debido a lo anteriormente expuesto se propone utilizar dicha tecnología para generar los artefactos necesarios para el diseño de la solución propuesta.

1.4.3. Herramienta CASE para UML:

Las herramientas de Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés) son usadas como un conjunto de programas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un Software, utilizando UML. A continuación se exponen las características de las herramientas CASE a utilizar.

Visual Paradigm en su versión 8.0 es herramienta CASE que propicia un conjunto de ayudas para el desarrollo de programas informáticos dando soporte al modelado visual con UML, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.(Oscar, 2013)

Se propone utilizar debido a que ofrece:

- Entorno de creación de modelos conformes a UML.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales Entornos de Desarrollo Integrado (IDE).
- Disponibilidad en múltiples plataformas.
- Extensible mediante desarrollo de nuevos Módulos (plug-ins).

1.4.4. Lenguaje de programación:

Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes. Tiene la capacidad de especificar, de forma precisa, cuáles son los datos que debe trabajar un equipo informático, de qué modo deben ser conservados o transferidos dichos datos y qué instrucciones debe poner en marcha la computadora ante ciertas circunstancias.(Stroustrup, 1986)

Lenguaje de Programación C++ en su versión 98 es un lenguaje imperativo, orientado a objetos, derivado de C. Al igual que C; C++ está muy ligado al hardware subyacente, manteniendo una considerable potencia para la programación a bajo nivel; pero se le han añadido elementos que permiten también un estilo de programación con alto nivel de abstracción. Debido a esto; C++ brinda la posibilidad de crear clases,

plantillas, sistema de espacios de nombres y funciones en línea, posee un mecanismo para el manejo de excepciones, permite la sobrecarga de operadores y utiliza operadores para el manejo de memoria. Este lenguaje de programación esta estandarizado por la Organización Internacional de Estándares (ISO, por sus siglas en inglés) y cuenta con una biblioteca estándar de alta calidad. (Stroustrup, 1986)

1.4.5. Marco de trabajo de desarrollo:

Un marco de trabajo es una abstracción de código común que provee funcionalidades genéricas que pueden ser utilizadas para desarrollar aplicaciones de manera rápida, fácil, modular y sencilla, ahorrando tiempo y esfuerzo. Los marco de trabajo de desarrollo son estructuras conceptuales y tecnológicas con soporte definido, normalmente con artefactos o Módulos de Software concretos; en base a la cual, otro proyecto de software puede ser más fácilmente desarrollado y organizado. Los de desarrollo son diseñados con la intención de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requisitos de software que tratando con los detalles de bajo nivel, para proveer un sistema funcional.(Távarez, 2014)

Marco de trabajo QT en su versión 5.2 es multiplataforma y se utiliza para el desarrollo de aplicaciones. Este está escrito en C++; sin embargo es posible utilizarlo con otros lenguajes como CSharp, PHP, Python, y Ruby. Qt ofrece bibliotecas de código para: creación de interfaces gráficas de usuario, acceso a bases de datos, manipulación de contenido Lenguajes de Marcas Extensible (XML, por sus siglas en inglés), comunicación en red, visualización con OpenGL, entre otras. Además; extiende el lenguaje de programación C++, a través de macros y meta-información, agregando nuevas características como: el bucle foreach, la sentencia forever e introspección.(Nokia, 2016)

Se propone utilizar el lenguaje de programación C++ y el marco de trabajo QT debido a que se podría reutilizar componentes y funcionalidades del HMI QT existente en el centro, además debido a que ambos son multiplataforma sería posible solventar el problema de la investigación.

1.4.6. Módulo de red de QT:

La clase **QNetworkAccessManager** permite a la aplicación para enviar las solicitudes de red y recibir respuestas, contiene la configuración y los ajustes para las solicitudes que envía, proxy y caché, así como las señales relacionados con esas emisiones y señales de respuesta. Realiza una transmisión asíncrona y encola las peticiones recibidas ,la cantidad de peticiones atendidas en paralelo depende del protocolo, en la actualidad el protocolo Protocolo de Transferencia de Hipertexto (HTTP, por sus siglas en inglés) en plataformas de escritorio ejecutan hasta 6 solicitudes para una combinación puerto anfitrión.(Ltd, 2015a)

1.4.7. Entorno de Desarrollo Integrado (IDE):

El Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) es un programa informático compuesto por herramientas de programación como; un editor de código, un compilador, un depurador y un constructor

de interfaz gráfica que permite desarrollar un software utilizando un lenguaje específico. A continuación se describe el IDE a emplear.

QtCreator en su versión 3.0 es de código abierto y multiplataforma creado por la empresa Trolltech para el desarrollo de aplicaciones utilizando las bibliotecas de Qt. Ofrece múltiples herramientas entre las que se encuentran: editor de código fuente para los lenguajes de programación C++ y JavaScript, diseñador de interfaces gráficas de usuario, ayuda sensible al contexto y depurador de código fuente. Además QtCreator puede ser integrado con sistemas de control de versiones y presenta una gran integración con el marco de trabajo Qt. Brinda completamiento de código para las clases pertenecientes a la biblioteca Qt y también dispone de un sistema de ayuda extenso y viendo documentado.(Ltd, 2015b)

1.4.8. Lenguaje para la transferencia de datos:

XML utilizado para almacenar datos en forma legible. Proviene del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que Lenguaje de Marcado de HiperTexto (HTML, por sus siglas en inglés) es a su vez un lenguaje definido por SGML) para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información. Propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto y hojas de cálculo.(Bray; Paoli; Sperberg-McQueen; Maler y Yergeau, 1998)

Notación de Objetos Javascript (JSON, por sus siglas en inglés) es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en JavaScript asíncrono y XML (AJAX, por sus siglas en inglés). Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador sintáctico de JSON. En JavaScript, un texto JSON se puede analizar fácilmente usando la función eval(), lo cual ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX.(JSON, s.f.)

Se propone utilizar JSON como lenguaje de transferencia de datos JSON debido a que su formato es sumamente simple, posee una alta velocidad de procesamiento alta y sus archivos son de menor tamaño.

1.5. Conclusiones Parciales:

En el presente capítulo se describieron los sistemas SCADA, el sistema SCADA SAINUX y su servidor, se analizaron los sistemas existentes HMI Qt y HMI Web. Después de haber analizado las metodologías, herramientas y tecnologías que hoy en día enriquecen a la informática se decidió utilizar para guiar el proceso de desarrollo de la solución propuesta la metodología AUP-UCI. Entre las tecnologías y las herramientas existentes se seleccionó el lenguaje de modelado el UML y el Visual Paradigm como herramienta CASE ,el lenguaje de programación C++ usando el marco de trabajo Qt siendo utilizado el entorno de desarrollo el

entorno de desarrollo integrado Qt Creator. Se definió JSON como lenguaje de transferencia de datos con el Servidor de HMI de SAINUX.

CAPÍTULO 2

Análisis y diseño del sistema.

Introducción.

En el presente capítulo tiene como objetivo realizar los procesos de análisis y diseño de la solución propuesta; el cual será guiado por la metodología de desarrollo AUP-UCI. Se propone el modelo de dominio donde se describen las entidades que intervienen con el objetivo de facilitar la comprensión de los principales conceptos que se utilizarán en el proceso de negocio. Se exponen los artefactos que describen el flujo normal de eventos que ocurren en el sistema, se realiza una descripción de la solución propuesta, planteando los requisitos funcionales y no funcionales, además de las historias de usuarios del sistema.

2.1. Análisis de la Solución.

La presente investigación está orientada al desarrollo de una aplicación de escritorio, capaz de conectarse con la capa de comunicaciones del SCADA SAINUX, a través de la Capa de Servicios Web para visualizar de forma tabulada todos los recursos asociados al módulo de adquisición a través de sumarios, los cuales son un conjunto de elementos representados en tabla que posee las acciones de filtrado y paginado. En el sistema se visualizan los sumarios de puntos analógicos, puntos digitales, subcanales, dispositivos y el ampliado y reducido de alarmas, estas últimas se encuentran ordenadas de acuerdo a la interfaz que la visualice, en el sumario ampliado por criticidad, mientras en el sumario reducido por la estampa de tiempo en que se produjo, además el sistema debe realizar una notificación sonora del estado en que se encuentran las alarmas del SCADA SAINUX. El sistema debería permitir ejecutar acciones de control sobre los dispositivos y mostrar los detalles de estos mismos y los subcanales, limitar a una única instancia las vistas de tipo sumario.

2.2. Modelo de Dominio.

El modelo de dominio es la representación visual de los conceptos u objetos más importantes de un negocio, sus características y las relaciones entre dichos conceptos. Permite comprender el dominio del

problema y para establecer conceptos comunes.(Jacobson; Booch y Rumbaugh, 2000)

En la solución propuesta el usuario interactúa con el Cliente de Escritorio el cual representa la información de forma tabular mediante Sumarios que pueden ser de Alarmas, Puntos Analógicos, Puntos Digitales, Dispositivos y Subcanales.

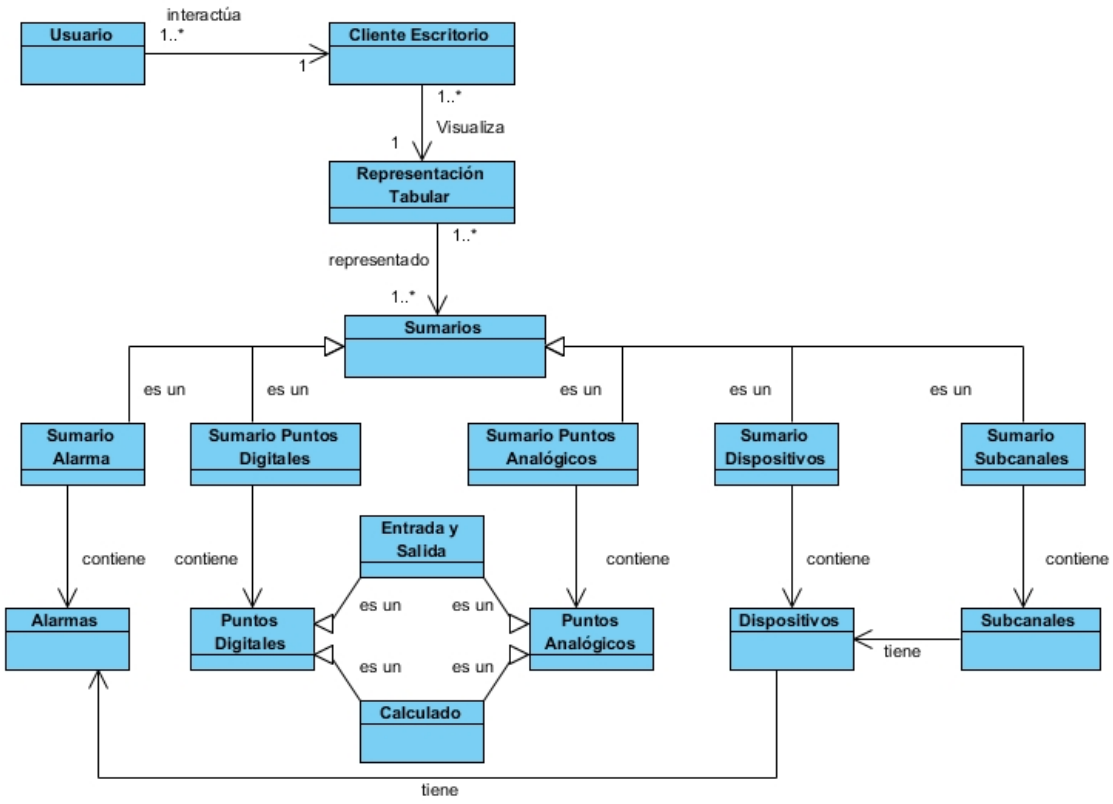

Figura 2.1. Modelo de Dominio propuesto

2.2.1. Descripción de los conceptos del dominio.

Conceptos del dominio	Descripción
Usuario	Persona que interactúa con el cliente de escritorio.
Cliente Escritorio	Herramienta encargada de la visualización de los datos.
Representación Tabular	Representación de la información en forma de tuplas.
Sumarios	Conjunto de elementos representados en tabla, y posee las siguientes acciones: filtro y paginado.
Alarmas	Es la capacidad de reconocer eventos excepcionales dentro del proceso y reportar estos eventos. Las alarmas son reportadas basadas en límites de control preestablecidos.
Puntos Analógicos	Representan las variables a monitorizar y controlar en el proceso, y se usan para medir valores de unidades de ingeniería como pueden ser presión o temperatura.
Puntos Digitales	Representan las variables a monitorizar y controlar en el proceso, representan valores binarios, por ejemplo para monitorizar el estado de un interruptor (abierto o cerrado, encendido o apagado).
Dispositivos	Son equipos electrónicos que pueden ser autómatas, Controlador Lógico Programable (PLC, por sus siglas en inglés), reguladores autónomos, sensores inteligentes, controladores, etc. Estos son el primer eslabón en la adquisición de los datos. A estos se le asocian puntos previamente configurados, cuyos valores se obtendrán mediante estos equipos.
Subcanales	Representan el protocolo de comunicación mediante el cual se conectará uno o varios dispositivos.
Sumario Alarma	Sumario que muestra un conjunto de alarmas.
Sumario Puntos Analógicos	Sumario que muestra un conjunto de puntos analógicos.
Sumario Puntos Digitales	Sumario que muestra un conjunto de puntos digitales.
Sumario Dispositivos	Sumario que muestra un conjunto de dispositivos.
Sumario Subcanales	Sumario que muestra un conjunto de subcanales.
Entrada y Salida	Tipo de punto que obtiene su valor de los dispositivos de campo.
Calculado	Tipo de punto que su valor se obtiene mediante la evaluación de una función matemática.

Tabla 2.1. Descripción de los conceptos del dominio.

2.3. Especificación de requisitos.

Conocer los requisitos del sistema es el primer paso en el desarrollo de su implementación, de esta manera se reducirá el número de cambios que habrá que realizar en el producto debido a los cambios en sus requisitos. A partir del modelo del dominio presentado se realizó el levantamiento de los requisitos, los cuales están divididos en dos tipos: los funcionales y los no funcionales. A continuación se especifican los requisitos definidos para el desarrollo del sistema propuesto.

2.3.1. Requisitos funcionales.

Los Requisitos Funcionales (RF) son capacidades o condiciones que el sistema debe cumplir para que las peticiones del usuario queden satisfechas.(Pressman, s.f.)

Teniendo en cuenta las funcionalidades que el sistema debe proveer se especificaron los siguientes RF:

RF1: Iniciar la aplicación.

RF2: El sistema debe permitir visualizar datos de usuario.

RF3: Interacción con el módulo de Seguridad.

- **RF3.1:** El sistema debe permitir autenticar usuario con perfiles.
- **RF3.2:** El sistema debe permitir cambiar contraseña.
- **RF3.3:** El sistema debe permitir renovar sesión.
- **RF3.4:** El sistema debe permitir cerrar sesión.

RF5: El sistema debe permitir visualizar sumario reducido de las alarmas.

RF6: El sistema debe permitir visualizar sumario de alarmas.

RF7: El sistema debe permitir visualizar sumario de puntos analógicos.

RF8: El sistema debe permitir visualizar sumario de puntos digitales.

RF9: El sistema debe permitir visualizar sumario de dispositivos.

RF10: El sistema debe permitir visualizar sumario de subcanales.

RF11: El sistema debe permitir realizar paginado de sumarios.

RF12: El sistema debe permitir filtrar sumarios.

RF13: El sistema debe permitir visualizar detalle de dispositivos.

RF14: El sistema debe permitir visualizar detalle de subcanales.

RF15: El sistema debe permitir ejecutar acciones sobre dispositivos.

- **RF15.1:** El sistema debe permitir ejecutar en barrido.
- **RF15.2:** El sistema debe permitir ejecutar fuera de barrido.

RF16: El sistema debe realizar la notificación sonora de la existencia de alarma acorde con el SCADA SAINUX.

RF17: El sistema debe permitir visualizar cada vista de sumario como máximo una sola vez.

2.3.2. Requisitos no funcionales.

Los Requisitos No Funcionales (RNF) son propiedades o cualidades que el producto debe poseer, debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Existen múltiples categorías para clasificar los requisitos no funcionales, siendo las siguientes las más representativas para el sistema propuesto.(ibíd.)

Usabilidad

RNF1: El sistema debe poder ser usado por cualquier persona que tenga conocimientos de sistemas SCADA.

Confiabilidad

RNF2: El sistema debe ser capaz de manejar los errores que ocurran en la conexión con la Capa de Servicios Web y recuperarse.

Interfaz de Usuario

RNF3: Forma de interacción de la interfaz: Basada en ventanas.

RNF4: El sistema debe contar con la política marcaria XEDRO.

RNF5: El texto del sistema debe ser en idioma español, tener una apariencia uniforme y seguir las normas de la política marcaria XEDRO.

Software

RNF6: Implementación con tecnologías libres.

RNF7: Sistema operativo GNU/Linux en su distribución Debian, específicamente la versión 8.

RNF8: Sistema operativo Microsoft Windows, específicamente la versión 10.

Hardware

RNF9: Como requisitos mínimos de hardware es necesario contar con memoria RAM 2GB y un microprocesador Intel(R) Core(Tm) 2 Duo.

2.4. Historias de Usuarios.

Es la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.(ibíd.)

Tabla 2.2. Historia de usuario # 1

Historia de usuario	
Número: 1	**Nombre:** Iniciar Aplicación
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta

Continúa en la próxima página

Tabla 2.2. Continuación de la página anterior

Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Alik del Risco	
Descripción: Iniciar ventana principal maximizada, en la cual se encuentra inicialmente activa la opción de autenticar.	

La descripción de las demás historias de usuario se encuentra en el anexo B.2.

2.5. Fase de Planificación

Una vez definida cada una de las historias de usuarios se pasó a definir la prioridad de cada una de ellas así como la estimación de esfuerzo necesario para realizar cada una de ellas. La planificación se realizó basándose en el tiempo o el alcance. Al planificar por tiempo, se multiplicó el número de iteraciones por la velocidad del proyecto, determinándose cuantos puntos se pueden completar. Al planificar según el alcance, se dividió la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

2.5.1. Estimación de esfuerzos por historia de usuario.

La siguiente tabla muestra estimación de esfuerzo para cada una de las Historia de Usuarios (HU) definidas en el desarrollo de la solución propuesta.

Historias de usuario	Puntos de estimación
Iniciar Aplicación.	1 Semana.
Visualizar datos de usuario.	0.5 Semana.
Autenticar usuario con perfiles.	1 Semana.
Cambiar contraseña.	1 Semana.
Renovar sesión.	1 Semana.
Cerrar sesión.	0.5 Semana.
Visualizar sumario reducido de las alarmas.	1 Semana.
Visualizar sumario de alarmas.	1 Semana.
Visualizar sumario de puntos analógicos.	1 Semana.
Visualizar sumario de puntos digitales.	1 Semana.
Visualizar sumario de dispositivos.	1 Semana.
Visualizar sumario de subcanales.	1 Semana.
Realizar paginado de sumarios.	0.5 Semana.
Filtrar sumarios	0.5 Semana.
Visualizar detalle de dispositivos.	1.5 Semana.
Visualizar detalle de subcanal.	1.5 Semana.
Ejecutar en barrido.	1.5 Semana.
Ejecutar fuera de barrido.	1.5 Semana.
Notificar de forma sonora la existencia de alarma acorde con el SCADA SAINUX.	0.5 Semana.
Visualizar cada vista de sumario como máximo una sola vez.	0.5 Semana.

Tabla 2.3. Estimación de esfuerzos.

2.5.2. Plan de Iteraciones.

Una vez definidas las HU y estimado el esfuerzo propuesto para la realización de cada una de ellas, se realizó el sistema en siete iteraciones, las cuales se describen a continuación de manera más detallada:

- **Iteración I:** Esta iteración tiene como objetivo realizar las HU 1,2,3 y 4 las cuales son las referentes a iniciar la aplicación, visualizar los datos de usuario, autenticar usuario con perfiles y cambiar la contraseña.
- **Iteración II:** Esta iteración tiene como objetivo realizar las HU 5,6 y 7 las cuales son las referentes a renovar y cerrar sesión, además visualizar sumario reducido de las alarmas.
- **Iteración III:** Esta iteración tiene como objetivo realizar las HU 8,9 y 10 las cuales son las referentes a visualizar los sumarios de alarmas, puntos analógicos y digitales.

- **Iteración IV:** Esta iteración tiene como objetivo realizar las HU 11,12,13 y 14 las cuales son las referentes a visualizar los sumarios de dispositivos y de subcanales, además de realizar el filtrado y paginado de los sumarios.
- **Iteración V:** Esta iteración tiene como objetivo realizar las HU 15 y 16 las cuales son las referentes a visualizar los detalles de dispositivos y subcanales.
- **Iteración VI:** Esta iteración tiene como objetivo realizar las HU 17 y 18 las cuales son las referentes a ejecutar acciones de control sobre los dispositivos.
- **Iteración VII:** Esta iteración tiene como objetivo realizar las HU 19 y 20 las cuales son las referentes notificar de forma sonora la existencia de alarma y visualizar cada vista de sumario como máximo una sola vez.

2.5.3. Plan de duración de las iteraciones.

Después de realizados la estimación de esfuerzo y el plan de iteraciones se crea el plan de duración de las iteraciones el cual tiene como objetivo fundamental mostrar la duración de cada iteración, así como el orden en que serán implementadas las HU en cada una de las mismas según la prioridad asignada por el cliente.

Iteración	Historias de Usuario	Duración total de las iteraciones (semanas)
Iteración I	Iniciar Aplicación. Visualizar datos de usuario. Autenticar usuario con perfiles. Cambiar contraseña.	3.5
Iteración II	Renovar sesión. Cerrar sesión. Visualizar sumario reducido de las alarmas.	2.5
Iteración III	Visualizar sumario de alarmas. Visualizar sumario de puntos analógicos. Visualizar sumario de puntos digitales.	3
Iteración IV	Visualizar sumario de dispositivos. Visualizar sumario de subcanales. Realizar paginado de sumarios. Filtrar sumarios.	3
Iteración V	Visualizar detalle de dispositivos. Visualizar detalle de subcanal.	3
Iteración VI	Ejecutar en barrido. Ejecutar fuera de barrido.	3
Iteración VII	Notificar de forma sonora la existencia de alarma acorde con el SCADA SAINUX. Visualizar cada vista de sumario como máximo una sola vez.	1

Tabla 2.4. Duración de las iteraciones.

2.6. Diagrama de Paquetes

Diagrama de paquetes en el UML representa las dependencias entre los paquetes que componen un modelo. Es decir, muestra cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre esas agrupaciones.(Ambler, 2005) Debido a que el diagrama de paquetes es muy extenso se divide por módulos. A continuación se muestra el diagrama de paquetes perteneciente al paquete Vista, el cual esta compuesto por View, Model, Controller y son los encargados de agrupar las vistas, modelos y las clases controladoras respectivamente, dentro del paquete View se encuentran los paquetes asociados a los distintos tipos de sumarios, de seguridad y vistas del sistema.

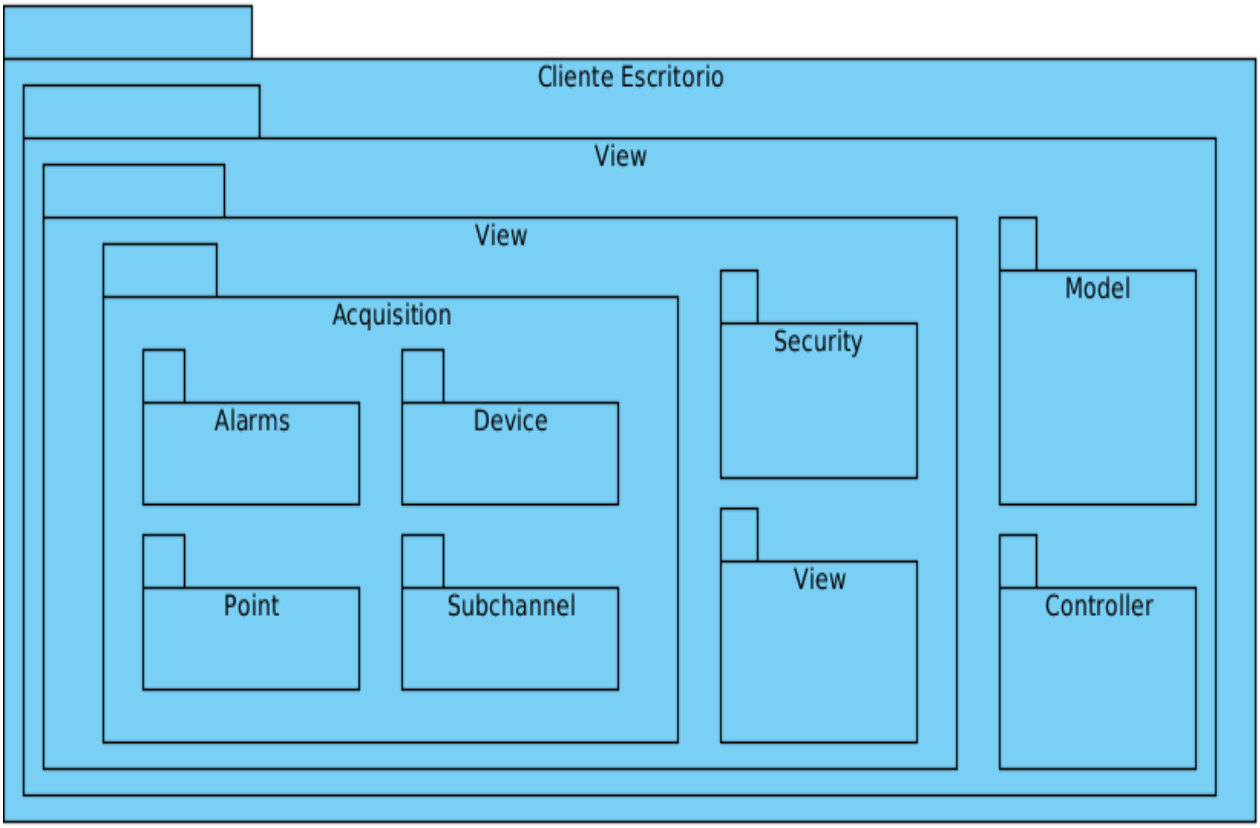

Figura 2.2. Diagrama de Paquetes perteneciente módulo View

Los restantes diagramas de paquetes se encuentran en el anexo B.1.

2.7. Diagrama de Secuencia

Para representar las interacciones entre los objetos en el sistema se utilizó los diagramas de secuencia. Además contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario y mensajes intercambiados entre los objetos.(RHRJJV Erich Gamma, 1994) A continuación se muestra el diagrama de secuencia correspondiente a la HU Autenticar usuario con perfiles:

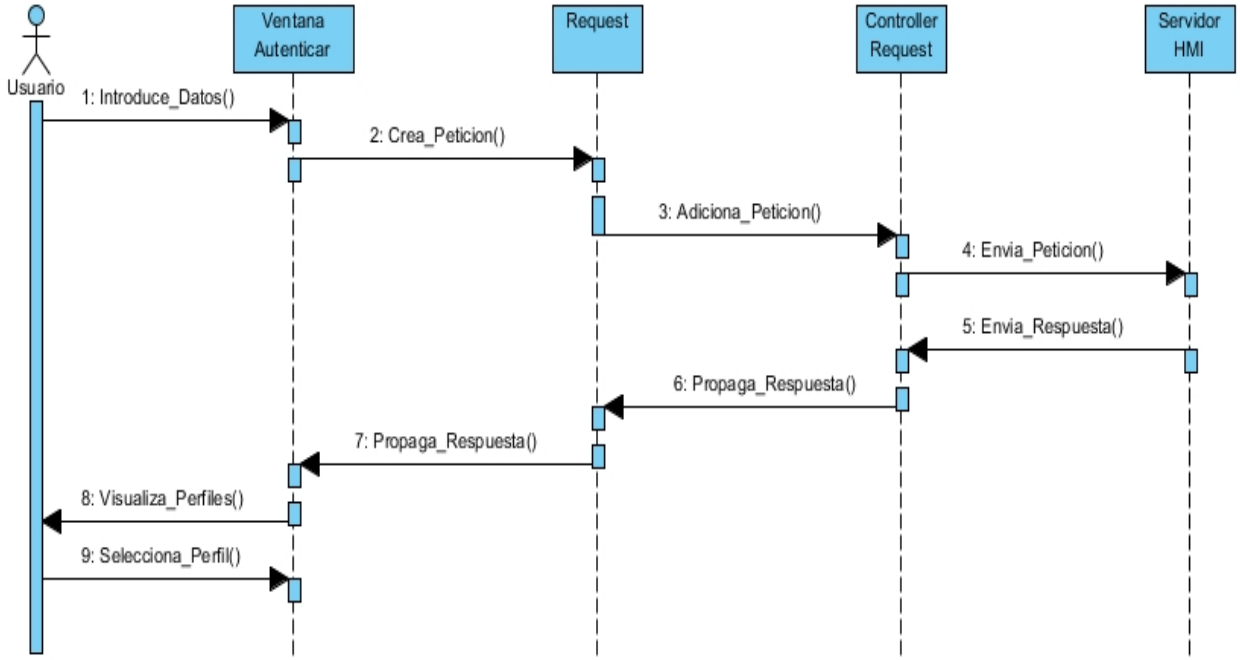

Figura 2.3. Diagrama de Secuencia Autenticar usuario con perfiles

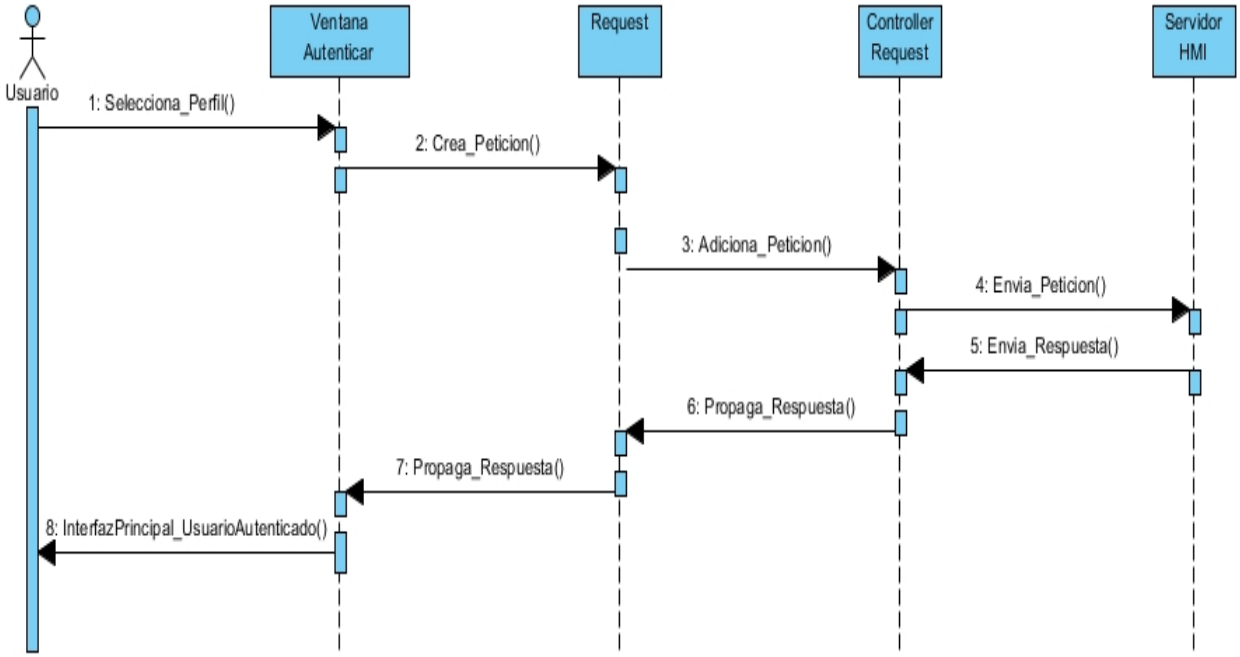

Figura 2.4. Diagrama de Secuencia Autenticar usuario con perfiles

Los restantes diagramas de secuencia se encuentran en el anexo B.3.

2.8. Modelo de Clases.

El diagrama de clases expresa la estructura u organización del software en términos de las clases. Es un reflejo abstracto de los componentes y las relaciones entre ellos. Los diagramas de clase son el pilar básico del modelado con UML, siendo utilizados tanto para mostrar lo que el sistema puede hacer, como para mostrar cómo puede ser construido. Cuando se crea un diagrama de clases, se está modelando una parte de los elementos y relaciones que configuran la vista de diseño del sistema.(Stevens; Pooley; Alarcón; Ó. S. Martínez y Sorrozal, 2007) A continuación se muestra los diagramas de clases correspondientes a la capa de Vista y Comunicación respectivamente:

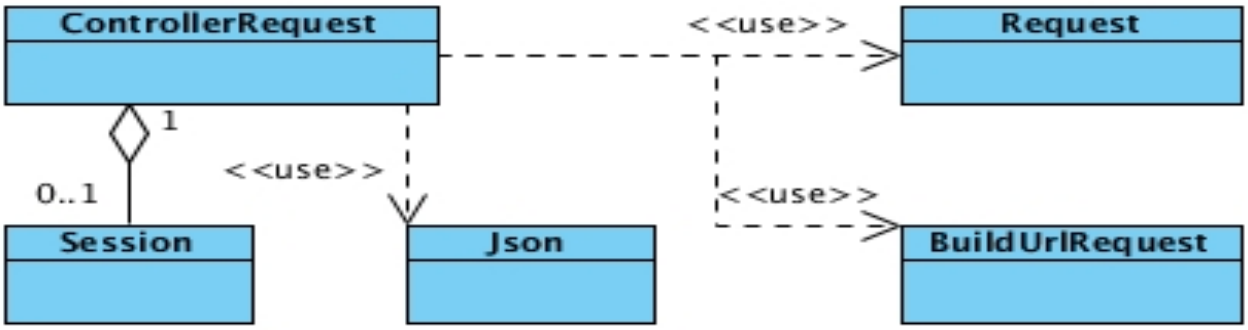

Figura 2.5. Modelo de Clases perteneciente a la Capa Comunicación

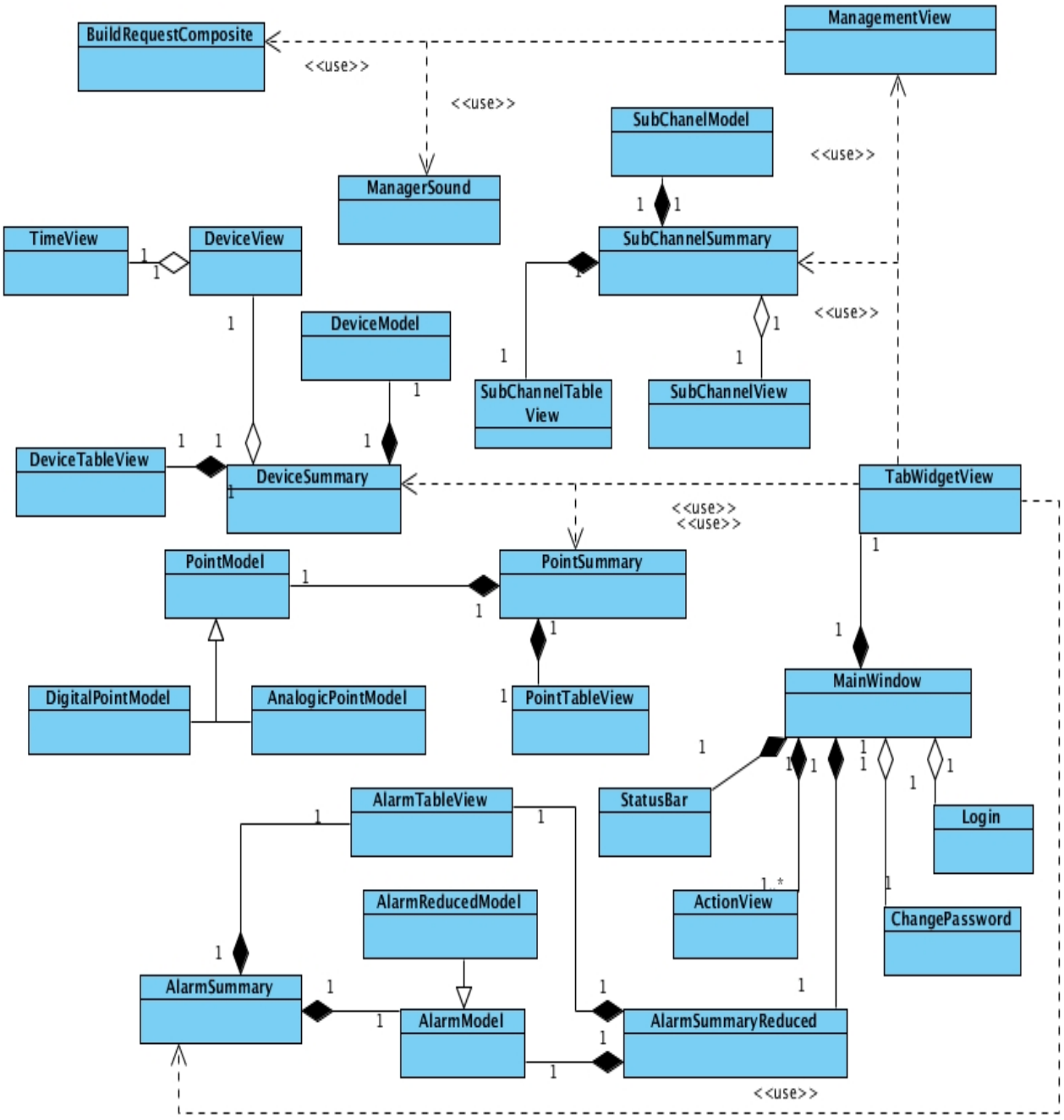

Figura 2.6. Modelo de Clases perteneciente a la Capa Vista

2.9. Patrones de Arquitectura.

Los patrones arquitectónicos, o patrones de arquitectura, son patrones de software que ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos com-

ponentes. Resuelven problemas arquitectónicos, adaptabilidad a requerimientos cambiantes, rendimiento, modularidad, acoplamiento, etc. La solución que plantea es la creación de patrones de llamadas entre objetos (similar a los patrones de diseño), decisiones y criterios arquitectónicos, empaquetado de funcionalidad. Este tipo de patrones se utilizan en la fase de desarrollo, en el diseño inicial. Los beneficios de la utilización de dichos patrones van desde la imposición de decisiones tempranas en el desarrollo hasta la reutilización.(Perry y Wolf, 1992)

La arquitectura del sistema está compuesta por 2 capas, la capa de Comunicación y la de Vista. La primera capa se encarga de construir y ejecutar los distintos tipos de peticiones realizadas por los usuarios , y la segunda de manejar la lógica del sistema y visualizar los datos. Internamente la capa de Vista se usa el patrón de arquitectura Modelo-Vista propuesto por el marco de trabajo Qt, el cual se divide en 3 grupos:

Modelos: Interfaz que es usada por vistas y delegados para acceder a los datos.

Vistas: Provee una interfaz estándar para interoperar con los modelos.

Delegados: Facilitan la visualización, edición y filtrado de los datos del modelo que utilizan las vistas.

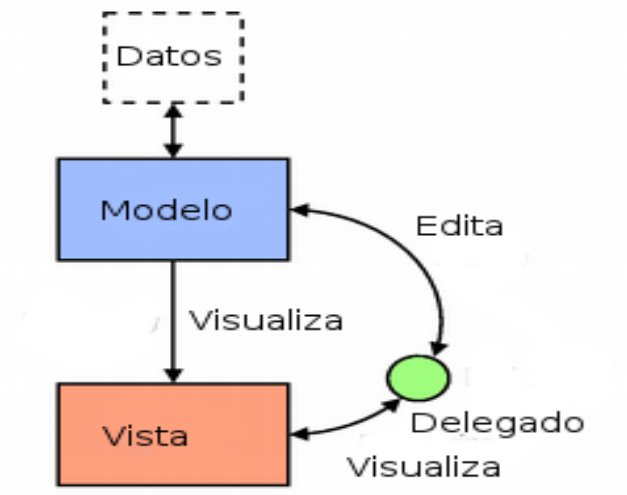

Figura 2.7. Patrón de Arquitectura Modelo-Vista de QT

Dicho patrón se emplea cuando se visualiza todos los tipos de sumarios, para mostrar el sumario de alarma el modelo sería la clase **AlarmModel**, la vista **AlarmTableView** y el delegado la clase **Paginator**.

2.10. Patrones de Diseño.

Los patrones de diseño expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software. Los patrones de diseño expresan un esquema organizativo estructural fundamental para sistemas de software. El uso de estos patrones proporciona una estructura conocida por todos los programadores, de manera que la forma de trabajar no resulte distinta entre los mismos. Así la incorporación de un nuevo programador, no requerirá conocimiento de lo realizado anteriormente por otro.(Freeman;

Robson; Bates y Sierra, 2004). Estos se pueden clasificar en:

- **Comportamiento:** Los patrones de comportamiento están relacionados con los algoritmos y la asignación de responsabilidades entre los objetos. Son utilizados para organizar, manejar y combinar comportamientos.
- **Estructurales:** Los patrones estructurales se ocupan de cómo las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades.
- **Creacionales:** Los patrones creacionales abstraen el proceso de creación de instancias y ocultan los detalles de cómo los objetos son creados o inicializados.

En solución propuesta se utilizan los siguientes patrones de diseño:

Instancia Única: Asegura que una clase tenga solamente una única instancia y provee un punto de acceso global a ella. Se utiliza en la clase **ControllerRequest** de la capa de Comunicación y pertenece al grupo de patrones creacionales.

Método de Plantilla: Define el esqueleto de un algoritmo en un método, difiriendo algunos pasos para subclases, además redefine ciertos pasos de un algoritmo sin cambiar su estructura. Se utiliza en el método **update** la clase **IView** de la capa de Vista y y pertenece al grupo de patrones comportamiento.

Fachada: Proporciona una interfaz unificada a un conjunto de interfaces en un subsistema. Define una interfaz de nivel superior haciendo al subsistema fácil de usar. Se utiliza en la clase **Request** de la capa de Comunicación y pertenece al grupo de patrones estructurales.

Fábrica Abstracta: Provee una interfaz para crear familias de objetos relacionados o dependientes entre ellos sin especificar una clase en concreto. Se utiliza en la clase **BuildUrlRequest** de la capa de Comunicación y pertenece al grupo de patrones creacionales.

2.11. Conclusiones Parciales.

Con el estudio realizado se pudo llegar a una comprensión clara y detallada de la solución propuesta, definiendo el modelo de dominio, permitiendo la identificación correcta, completa y consistente de los requisitos funcionales y no funcionales, de ellos 17 funcionales representados en historias de usuario y 9 no funcionales. La especificación y descripción de cada una de las historias de usuario sirvió de guía para definir la estructura del sistema mediante el diagrama de paquetes para representar las dependencias entre los paquetes que componen el modelo, los diagramas de clases que expresa la estructura u organización del software en términos de las clases, la arquitectura la cual está compuesta por 2 capas, la capa de Comunicación y la de Vista, en la capa de Vista se usa internamente el patrón de arquitectura Modelo-Vista propuesto por el marco de trabajo Qt y los patrones de diseño para solucionar los problemas más comunes; todos los elementos antes mencionados constituyen una entrada fundamental para el desarrollo de la solución.

CAPÍTULO 3

Implementación y prueba

3.1. Introducción.

En el presente capítulo se analizan los elementos necesarios para proceder a la implementación del sistema y se verifica el cumplimiento de los requisitos funcionales mediante las pruebas de software. Se presenta el Modelo de Implementación que expone mediante diagramas de componentes como las clases del diseño se implementan en términos de componentes y sus relaciones. Además, este capítulo tiene como objetivo, comprobar que el Cliente de Escritorio funcione correctamente, aplicando distintos tipos de prueba para la detección de posibles errores, garantizando de esta forma la calidad requerida.

3.2. Modelo de implementación.

El Modelo de Implementación describe como los elementos del Modelo de Diseño (ejemplo las clases), se implementan en términos de componentes, incluyendo ficheros de código fuente, ejecutables, librerías, tablas de base de datos o documentos. El Modelo de Implementación describe también como se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados. Además, el Modelo de Implementación es la entrada principal a la etapa de prueba.(Jacobson; Booch y Rumbaugh, 2000)

3.2.1. Diagrama de Componentes.

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, bibliotecas cargadas dinámicamente, etc. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente.(P. L. Torres, 2004)

A continuación, se presenta el Diagrama de Componentes correspondiente a la solución:

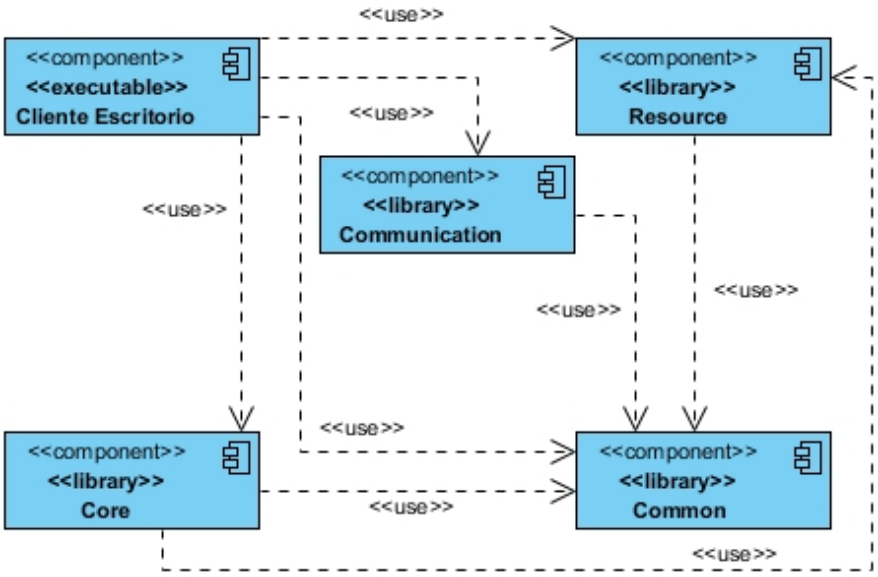

Figura 3.1. Diagrama de Componentes

3.2.2. Descripción del diagrama de componentes.

El Diagrama de Componentes está compuesto por los siguientes elementos:

Cliente Escritorio: Aplicación con la cual el usuario interactúa y ejecuta las distintas operaciones permitidas por el sistema.

Communication: Biblioteca encargada de establecer la comunicación entre el cliente y el servidor, además construye y ejecuta los distintos tipos de peticiones realizadas por los usuarios.

Common: Biblioteca que brinda un conjunto de funcionalidades comunes que son utilizadas en los distintos elementos debido a su alta utilidad.

Resource: Biblioteca que gestiona los recursos externos (imágenes, estilos, sonidos) necesarios para la aplicación.

Core: Biblioteca que provee las principales interfaces y vistas genéricas usadas en el sistema.

3.3. Modelo de Despliegue.

Con el objetivo de proveer una descripción de la distribución física del sistema se realiza el modelo de despliegue, mediante el cual se muestran las relaciones físicas de los distintos nodos que componen el sistema. Los diagramas de despliegue son los complementos de los diagramas de componentes que, unidos, proporcionan la vista de implementación del sistema. A su vez describen la topología del sistema, es decir la estructura de los elementos de hardware y el software que consumen. Además muestran las relaciones físicas

de los distintos nodos que componen un sistema.(Jacobson; Booch y Rumbaugh, 2000) A continuación se muestra el modelo de despliegue del sistema:

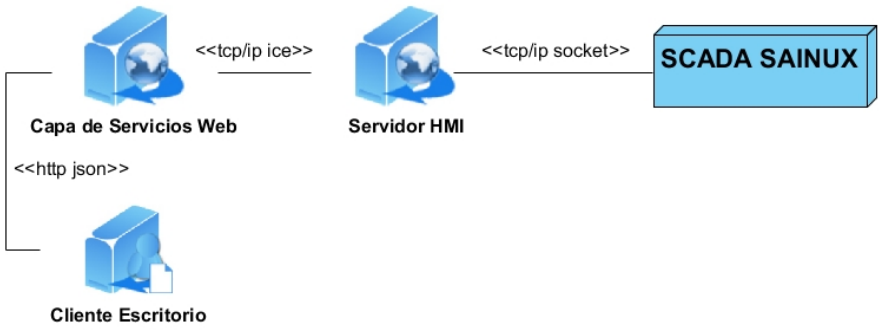

Figura 3.2. Diagrama de Despliegue

El diagrama de despliegue de la solución esta compuesta por cuatro nodos:

- Cliente Escritorio: Aplicación de escritorio del ambiente de visualización del sistema SCADA SAINUX, el cual interactúa con el sistema del SCADA SAINUX a través de la Capa de Servicios Web.
- Capa de Servicios Web: Provee de una interfaz de comunicación mediante la cual diferentes clientes del ambiente de visualización del SCADA SAINUX independientemente de la plataforma en que estén desarrollados interactúan con el del Servidor HMI.
- Servidor HMI: Sistema brinda un conjunto de funcionalidades para la supervisión y control de procesos industriales desde clientes ligeros del ambientes visualización entre las que destacan: el envío de la información referente a los sumarios de puntos, alarmas, subcanales y dispositivos.
- SCADA SAINUX: El sistema SCADA SAINUX es un sistema distribuido, compuesto por diferentes subsistemas o módulos que trabajan de manera conjunta para llevar a cabo las tareas de supervisión, control y adquisición de datos. Ofrece estos datos a los restantes nodos que componen el diagrama de despliegue.

3.4. Estándares de codificación.

Uno de los instrumentos que facilitan la tarea de asegurar la calidad del software es la adopción de estilos y estándares de codificación. El uso de estos estándares tiene innumerables ventajas entre ellas lograr un estilo de código homogéneo asegurando su legibilidad y proveer una guía para el encargado del mantenimiento/actualización del sistema, con código claro y bien documentado. Además ayuda a mejorar el proceso de codificación haciéndolo en gran medida eficiente y en muchos casos reutilizables. A continuación se exponen los diferentes estilos de codificación que se utilizaron en la implementación del sistema:

- Los atributos de la clase comienzan con m seguido de guión bajo y el nombre de la variable. Ejemplo:

```
bool m_startSession;
```

- Las variables temporales de los métodos comienzan con t seguido del nombre de la variable. Ejemplo:

```
unsigned int tTimeSession=0;
```

- Los nombres de los métodos comienzan con letra minúscula. Ejemplo:

```
void createConnects();
```

- Los argumentos de los métodos comienzan guión bajo seguido de su nombre. Ejemplo:

```
void update(QJsonObject _data);
```

- Los nombres de los métodos o variables que estén compuestos por varias palabras se aplica el estilo de escritura de Camello. Ejemplo:

```
void addRequestSon(Request* _request);
```

3.5. Validación y pruebas.

El uso de las pruebas permite comprobar el funcionamiento de los códigos que se vayan implementando. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones. Las pruebas se pueden realizar basándose en dos esquemas diferentes: demostrar a través de pruebas de caja blanca que las operaciones internas se ajustan a lo especificado y que los componentes internos marchan bien, o mediante las pruebas de caja negra, conociendo la función del programa e intentar demostrar que las funciones están correctas.

3.5.1. Pruebas de Aceptación.

Las pruebas de aceptación son creadas en base a las historias de usuarios definidas por el cliente. Dichas pruebas son consideradas como pruebas de caja negra y los clientes son los responsables de verificar que el resultado de estas pruebas sean los correctos. Una HU puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar un correcto funcionamiento.(Gutiérrez; Escalona; Mejías y J. Torres, 2006)

Tabla 3.1. Prueba de aceptación # 1

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Iniciar Aplicación	
Descripción: Iniciar ventana principal ,en la cual solamente esta activa la opción de autenticar.	
Condiciones de ejecución: Sistema operativo GNU/Linux en su distribución Debian, específicamente la versión 8 ó sistema operativo Microsoft Windows, específicamente la versión 10.	
Pasos de ejecución: El usuario ejecuta la aplicación.	
Resultados esperados: En el sistema operativo Microsoft Windows 10 las dimensiones de la interfaz no ocupan toda la pantalla en la cual se visualiza, dicha no conformidad se corrige en la primera iteración destinada a la solución de dichos eventos. Logrando la correcta visualización la interfaz del sistema.	

Las restantes pruebas de aceptación se encuentran en el anexo C.0.1.

3.5.2. Pruebas de Rendimiento.

Las pruebas de rendimiento se realizan desde una perspectiva para determinar lo rápido que realiza una tarea un sistema en condiciones particulares de trabajo. También puede servir para validar y verificar otros atributos de la calidad del sistema, tales como la escalabilidad, fiabilidad y uso de los recursos. Las pruebas de rendimiento son un subconjunto de la ingeniería de pruebas, una práctica informática que se esfuerza por mejorar el rendimiento, englobándose en el diseño y la arquitectura de un sistema, antes incluso del esfuerzo inicial de la codificación.(Molyneaux, 2009)

Para la aplicación de la prueba de rendimiento, se hace un minucioso análisis del uso del procesador y consumo de memoria. A continuación se exponen las características de hardware, de software de la computadora donde se ejecutará el sistema y los resultados que arrojaron los diferentes factores a evaluar al iniciar la aplicación y después de ejecutarse 24 horas consecutivas.

Prestaciones de Hardware:

- Memoria RAM: 4GB DDR3 1600Mhz.
- Procesador: Intel Core i3-2120 CPU 3.30GHz x 4.

Sistema Operativo:

- Debian Wheezy 64 bits.
- Microsoft Windows 10 64 bits.

Sistema Operativo.	Consumo Memoria(MB)	Uso Procesador(%)
Microsoft Windows 10	80	6
Debian Wheezy	70	4

Tabla 3.2. Consumo de Memoria y uso de Procesador en distintos sistemas operativos al iniciar la aplicación.

Sistema Operativo.	Consumo Memoria(MB)	Uso Procesador(%)
Microsoft Windows 10	84	6
Debian Wheezy	72	4

Tabla 3.3. Consumo de Memoria y uso de Procesador en distintos sistemas operativos despues de 24 h de ejecución.

Los datos arrojados indican que es el sistema operativo Debian Wheezy posee el mejor rendimiento en cuanto la visualización de los datos pertenecientes al SCADA SAINUX.

De acuerdo con los resultados obtenidos, las pruebas se realizaron satisfactoriamente obteniéndose valores satisfactorios en concordancia con el tiempo de actualización que debe poseer el visualizador de escritorio para la representación de los procesos pertenecientes al SCADA SAINUX.

3.5.3. Resultados de las Pruebas.

Después de realizar las pruebas de aceptación y rendimiento se detectaron 8 no conformidades en total. Para solventarlas se realizó un plan compuestos por cuatro iteraciones, donde se detectaron 4 no conformidades en la primera, 2 en la segunda y tercera, y ninguna en la cuarta. Cada problema detectado en las iteraciones fue resuelto inmediatamente a raíz del trabajo continuo del desarrollador, comprobando asi el correcto funcionamiento del sistema. A continuación se representa lo expuesto anteriormente a través de la siguiente figura.

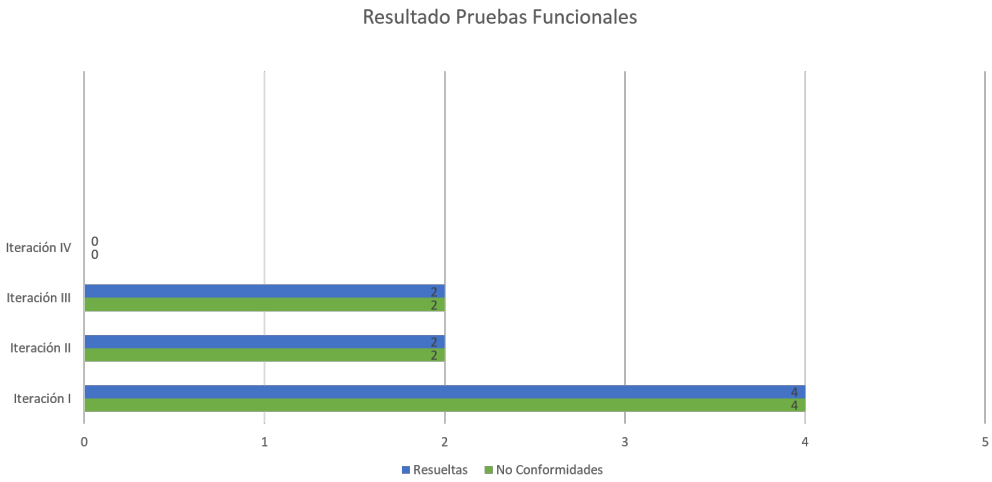

Figura 3.3. Resultado de las Pruebas Funcionales

3.6. Conclusiones Parciales.

En este capítulo se definió el diagrama de despliegue, el cual muestra los elementos necesarios para realizar el despliegue e instalación del sistema. Se define el modelo de implementación y el estándar de codificación a utilizar. Se desarrollaron las tareas correspondientes para dar solución a las historias de usuario. Además se ejecutaron las pruebas de aceptación y de rendimiento para detectar las no conformidades del sistema. A partir del resultado arrojado por las pruebas realizadas se llegó a la conclusión de que el Cliente de Escritorio se encuentra listo para su puesta en funcionamiento. Con la finalización de este capítulo se da por terminada la propuesta de solución del mecanismo a elaborar por el autor.

Conclusiones

Una vez concluida la presente investigación se logró:

- Se establecieron los fundamentos teórico-metodológicos para el desarrollo de sistemas informáticos SCADA como entorno de escritorio.
- Se desarrolló un cliente de escritorio del entorno de visualización que elevó la portabilidad de la Interfaz Hombre-Máquina del SCADA SAINUX.
- Las pruebas de aceptación y rendimiento validaron los resultados obtenidos.

Destacar que todo el desarrollo del trabajo se utilizó herramientas y tecnologías libres.

Recomendaciones

Luego de haber analizado los resultados del presente trabajo de diploma, se recomienda :

- Concluir las operaciones referentes al módulo de Adquisición:
  - Forzar y escribir en los puntos.
  - Inhibir, reconocer y silenciar las alarmas.
  - Visualización de los detalles de los puntos.
- Añadir las funcionalidades de visualización de la información referente al módulo Interfaz Hombre Máquina:
  - Visualización de los despliegues.
  - Visualización de los reportes.
- Visualización de la información referente al módulo Histórico.
- Para aumentar la seguridad en la comunicación con la Capa de Servicios Web se propone usar el protocolo https.

Acrónimos

AJAX JavaScript asíncrono y XML. 15
AUP Proceso Unificado Ágil. 10–12

CASE Ingeniería de Software Asistida por Computadora. 13, 15
CEDIN Centro de Informática Industrial. 1, 8–10

EC Entorno de Configuración. 1, 6
EV Entorno de Visualización. 1, 6

HMI Interfaz Hombre-Máquina. 1, 3–5, 7–10, 14, 15, 33
HTML Lenguaje de Marcado de HiperTexto. 15
HTTP Protocolo de Transferencia de Hipertexto. 14
HU Historia de Usuarios. 21–23, 25

IDE Entorno de Desarrollo Integrado. 14

JSON Notación de Objetos Javascript. 15

PLC Controlador Lógico Programable. 18

RF Requisitos Funcionales. 19
RNF Requisitos No Funcionales. 20
RUP Proceso Unificado Racional. 10

SAINUX Sistema de Automatización Industrial basado en GNU/LINUX. 1, 2, 8, 10, 15, 16, 33, 36
SCADA Control, Supervisión y Adquisición de Datos. 1–10, 15, 16, 20, 33, 36

UCI Universidad de las Ciencias Informáticas. 1, 9–12
UML Lenguaje Unificado de Modelado. 12, 13, 15, 24, 27

XML Lenguajes de Marcas Extensible. 14, 15

Referencias bibliográficas

Ambler, Scott W, 2005. *The Elements of UML (TM) 2.0 Style* (vid. pág. 25).

Automation, Inductive, 2013. *Inductive Automation Module Features*. Dirección: ⟨https://inductiveautomation.com/scada-software/scada-modules⟩ (vid. pág. 8).

Booch, Grady; Rumbaugh, James; Jacobson, Ivar; Martínez, José Sáez y Molina, Jesús J García, 1999. *El lenguaje unificado de modelado* (vid. pág. 13).

Bray, Tim; Paoli, Jean; Sperberg-McQueen, C Michael; Maler, Eve y Yergeau, François, 1998. Extensible markup language (XML). *World Wide Web Consortium Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210*, vol. 16 (vid. pág. 15).

Durán, María Elena Otero, 2009. TELENUL, SCADA de Supervisión y Telecontrol de Redes de Distribución, diseñado para recerradores y seccionalizadores NULEC. (Vid. pág. 8).

Freeman, Eric; Robson, Elisabeth; Bates, Bert y Sierra, Kathy, 2004. *Head first design patterns* (vid. pág. 30).

García, Darlin Mercedes Blanco, 2013. *Simulador de Variables Analógicas y Digitales para la Interfaz Hombre Máquina (HMI) del SCADA Guardián del Alba.* (Vid. pág. 6).

Gelbmann, Roland, 2002. *Evaluation and Comparison of CORBA (Object Request Broker) Implementations* (vid. pág. 9).

Gutiérrez, JJ; Escalona, MJ; Mejías, M y Torres, J, 2006. Pruebas del Sistema en Programación Extrema. *Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla* (vid. pág. 35).

Jacobson, Ivar; Booch, Grady y Rumbaugh, James, 2000. *El proceso unificado de desarrollo de software* (vid. págs. 18, 32, 34).

Jaime Fardales Pérez Yaneisy Hernández, Moisés Herrera Vázquez, 2008. Sistema Supervisor Guardián del ALBA. (Vid. pág. 8).

JSON. *2016*. Dirección: ⟨http://www.json.org/json-es.html⟩ (vid. pág. 15).

Ltd, The Qt Company, 2015a. Dirección: ⟨http://doc.qt.io/qt-5/⟩ (vid. pág. 14).

Ltd, The Qt Company, 2015b. Dirección: ⟨http://www.qt.io/ide/⟩ (vid. pág. 15).

Martínez, Yosvani Ramírez, 2013. *Mecanismo de incorporación de scripting al HMI del SCADA Guardián del ALBA.* (Vid. pág. 6).

MeasureSoft., 2009. Dirección: ⟨http://www.measuresoft.com/products/scadapro-server/⟩ (vid. pág. 7).

Molyneaux, Ian, 2009. *The Art of Application Performance Testing: Help for Programmers and Quality Assurance* (vid. pág. 36).

Moore, J Strother, 1994. A formal model of asynchronous communication and its use in mechanically verifying a biphase mark protocol. *Formal Aspects of Computing*, vol. 6, n.º 1, págs. 60-91 (vid. pág. 9).

Nokia, Qt, 2016. Dirección: ⟨http://www.qt.io/developers/⟩ (vid. pág. 14).

Oscar, S., 2013. *Visual Paradigm for Uml*. Url: ⟨https://books.google.com.cu/books?id=PDX-lgEACAAJ⟩. ISBN 9786139166534 (vid. pág. 13).

Penin, Aquilino Rodríguez, 2011. *Sistemas Scada* (vid. págs. 5, 6).

Perry, Dewayne E y Wolf, Alexander L, 1992. Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*, vol. 17, n.º 4, págs. 40-52 (vid. pág. 30).

Pressman, Roger. *S. 2002, Ingeniería de Software: Un enfoque práctico* (vid. págs. 20, 21).

Pro, MeasureSoft SCADA, 2016. Dirección: ⟨http://www.measuresoft.com/scadapro⟩ (vid. pág. 7).

RHRJJV Erich Gamma, Design Patterns, 1994. *Elements of Reusable Object-Oriented Software* (vid. pág. 26).

Sánchez, Tamara Rodríguez, 2014. Metodología de desarrollo para la Actividad productiva de la UCI. (Vid. pág. 12).

SERCONI, 2010. Ficha Técnica SCADA Eros. (Vid. pág. 8).

Siemens, 2016. Dirección: ⟨http://w3.siemens.com/mcms/human-machine-interface/en/visualization-software/scada/Pages/system-overview.aspx⟩ (vid. pág. 7).

Silva, Nuno Miguel Milhases da, 2012. Suporte à interoperabilidade entre o Automation Studio e Sistemas SCADA: tradução de sinópticos de XAML para SVG. (Vid. pág. 8).

Stevens, Perdita; Pooley, Rob; Alarcón, Marta Fernández; Martínez, Óscar Sanjuan y Sorrozal, Francisco Pérez, 2007. *Utilización de UML en Ingeniería del Software con Objetos y Componentes* (vid. pág. 28).

Stroustrup, Bjarne, 1986. *The C++ programming language* (vid. págs. 13, 14).

Távarez, David, 2014. Comparación de Frameworks en Javascript. Url: ⟨http://www.maestrosdelweb.com/comparacion-frameworks-javascript⟩ (vid. pág. 14).

Torres, Patricio Letelier, 2004. Desarrollo de Software Orientado a Objeto usando UML. *Universidad Politecnica de Valencia (UPV)–España* (vid. pág. 32).

Generado con LaTeX: 15 de junio de 2016: 9:55am

Apéndices

APÉNDICE A

Fundamentación Teórica

A.1. Conceptos Asociados

Entrevista al especialista del CEDIN Luis Andrés Valido Fajaro.

¿Cuál es el concepto asociado al término comando?

¿Qué es una acción operacional?

¿Cuál concepto se tiene en el centro sobre los sumarios?

¿Qué son los despliegues?

APÉNDICE B

Análisis y diseño del sistema.

B.1. Diagrama de Paquetes

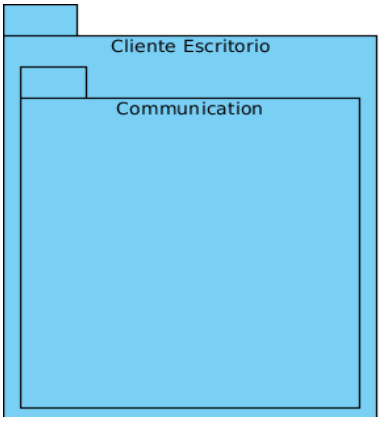

Figura B.1. Diagrama de Paquete de Comunicación

Diagrama de paquetes perteneciente al paquete Comunicación, el cual esta contiene los subpaquetes encargados de establecer la comunicación entre el cliente y el servidor, construir y ejecutar los distintos tipos de peticiones realizadas por los usuarios.

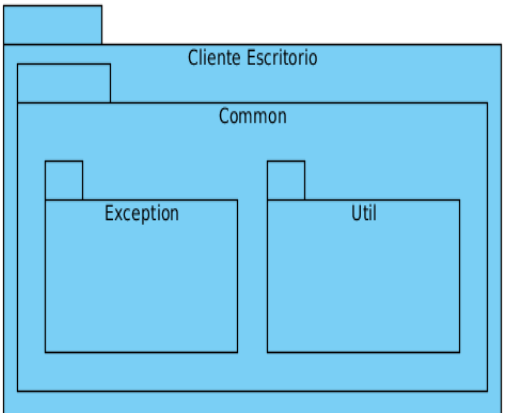

Figura B.2. Diagrama de Paquete Común

Diagrama de paquetes perteneciente al paquete Común, el cual esta compuesto por los subpaquetes Exception y Util, el primero encargado de la las excepciones del sistema y el segundo contiene los subpaquetes que son de uso común el los restantes paquetes que conforman el sistema, como por ejemplo las macros, patrón de diseño Instancia Única, temporizador, y validación de distintos de campos.

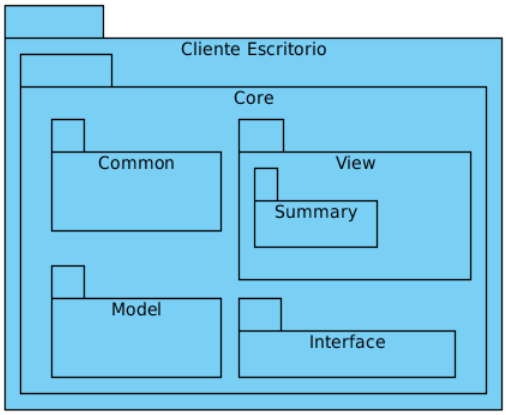

Figura B.3. Diagrama de Paquete Core

Diagrama de paquetes perteneciente al paquete Core, el cual esta compuesto por los subpaquetes View, Model, Interface, Common que proveen las principales interfaces, funcionalidades, vistas genéricas del sistema.

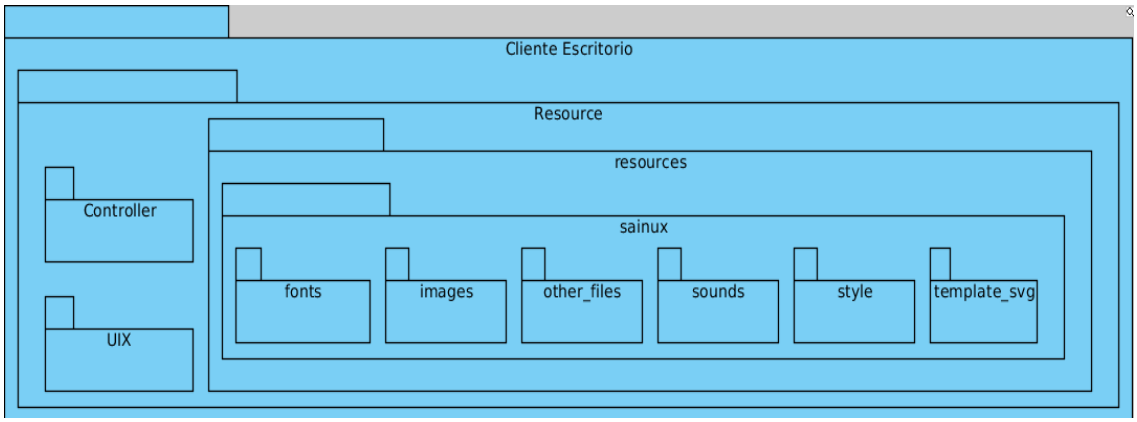

Figura B.4. Diagrama de Paquete Resource

Diagrama de paquetes perteneciente a la capa Resource, el cual esta compuesto por los subpaquetes Controller, UIX, Sainux, donde este último a su vez contiene fonts, images, other_files, sounds, style, template_svg que agrupan los ficheros segun indica su nombre. De modo general este paquete gestiona los recursos externos necesarios para la aplicación.

B.2. Historias de Usuarios.

Tabla B.1. Historia de usuario # 2

Historia de usuario	
Número: 2	**Nombre:** Visualizar datos de usuario.
Usuario: Usuario	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.5	**Iteración asignada:** 1
Programador responsable: Alik del Risco	
Descripción: El sistema debe visualizar los datos del usuario tales como el nombre de usuario y el tiempo restante de sesión.	

Tabla B.2. Historia de usuario # 3

Historia de usuario	
Número: 3	**Nombre:** Autenticar usuario con perfiles
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Alik del Risco	

Continúa en la próxima página

Tabla B.2. Continuación de la página anterior

Descripción: El sistema debe poseer una interfaz de autenticación para introducir su usuario,contraseña y una lista con sus perfiles asociados.

Tabla B.3. Historia de usuario # 4

Historia de usuario	
Número: 4	**Nombre:** Cambiar contraseña.
Usuario: Usuario	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Alik del Risco	
Descripción: El sistema muestra una interfaz que permite introducir su contraseña actual dos veces y la nueva.	

Tabla B.4. Historia de usuario # 5

Historia de usuario	
Número: 5	**Nombre:** Renovar sesión.
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Alik del Risco	
Descripción: Cuando el tiempo de sesión del usuario autenticado en el sistema termina y este tiene permiso para renovar la sesión, el sistema le pregunta a través de una ventana emergente si desea renovar o cerrar la sesión. Solo se podrá renovar la sesión una sola vez.	

Tabla B.5. Historia de usuario # 6

Historia de usuario	
Número: 6	**Nombre:** Cerrar sesión
Usuario: Usuario	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.5	**Iteración asignada:** 2
Programador responsable: Alik del Risco	
Descripción: El sistema permite cerrar la sesión si el usuario lo decide mediante una ventana emergente, o cuando su tiempo de sesión expire el sistema automáticamente cerraria la sesión.	

Tabla B.6. Historia de usuario # 7

Historia de usuario	
Número: 7	**Nombre:** Visualizar sumario reducido de las alarmas.
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Alik del Risco	
Descripción: El sistema debe mostrar siempre el sumario reducido de alarmas, mostrando las últimas 5 alarmas generadas por el sistema. Los campos a mostrar son los siguientes: fecha, hora, recurso, causa, tipo, descripción, prioridad, grupo, estado, número de ocurrencia.	

Tabla B.7. Historia de usuario # 8

Historia de usuario	
Número: 8	**Nombre:** Visualizar sumario de alarmas.
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Alik del Risco	
Descripción: El sistema debe permitir al usuario visualizar el estado de las alarmas, ordenas por criticidad del sistema. Los campos a mostrar son los siguientes: fecha, hora, recurso, causa, tipo, descripción, prioridad, grupo, estado, número de ocurrencia.	

Tabla B.8. Historia de usuario # 9

Historia de usuario	
Número: 9	**Nombre:** Visualizar sumario de puntos analógicos.
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Alik del Risco	
Descripción: El sistema debe permitir al usuario visualizar el estado de los puntos analógicos. Los campos a mostrar son los siguientes: nombre, descripción, valor actual, unidad de ingeniería, grupo, símbolos de calidad, tipo, dispositivo.	

Tabla B.9. Historia de usuario # 10

Historia de usuario	
Número: 10	**Nombre:** Visualizar sumario de puntos digitales.

Continúa en la próxima página

Tabla B.9. Continuación de la página anterior

Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Alik del Risco	
Descripción: El sistema debe permitir al usuario visualizar el estado de los puntos digitales. Los campos a mostrar son los siguientes: nombre, descripción, valor actual, grupo, símbolos de calidad, tipo, dispositivo.	

Tabla B.10. Historia de usuario # 11

Historia de usuario	
Número: 11	**Nombre:** Visualizar sumario de dispositivos.
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 4
Programador responsable: Alik del Risco	
Descripción: El sistema debe permitir al usuario visualizar el estado de los dispositivos. Los campos a mostrar son los siguientes: recurso, descripción , estado actual, tiempo de encuesta actual, estado anterior, tiempo de encuesta anterior.	

Tabla B.11. Historia de usuario # 12

Historia de usuario	
Número: 12	**Nombre:** Visualizar sumario de subcanales.
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 4
Programador responsable: Alik del Risco	
Descripción: El sistema debe permitir al usuario visualizar el estado de los subcanales. Los campos a mostrar son los siguientes: recurso, descripción, estado actual, tiempo de encuesta actual, estado anterior, tiempo de encuesta anterior.	

Tabla B.12. Historia de usuario # 13

Historia de usuario	
Número: 13	**Nombre:** Realizar paginado de sumarios.
Usuario: Usuario	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 0.5	**Iteración asignada:** 4
Programador responsable: Alik del Risco	

Continúa en la próxima página

Tabla B.12. Continuación de la página anterior

Descripción: El sistema debe permitir el paginado de los sumarios, de esta manera se dividen contenidos de gran longitud en varias páginas más cortas. La cantidad de datos a mostrar por página debe ser configurable.

Tabla B.13. Historia de usuario # 14

Historia de usuario	
Número: 14	**Nombre:** Filtrar sumarios.
Usuario: Usuario	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 0.5	**Iteración asignada:** 4
Programador responsable: Alik del Risco	
Descripción: El sistema debe permitir filtrar los sumarios en dependencia de los campos que muestre cada sumario ó de algún valor en específico, de esta manera se pueden realizar búsquedas por categorías.	

Tabla B.14. Historia de usuario # 15

Historia de usuario	
Número: 15	**Nombre:** Visualizar detalle de dispositivos.
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1.5	**Iteración asignada:** 5
Programador responsable: Alik del Risco	
Descripción: El sistema debe permitir al usuario visualizar los detalles del dispositivo seleccionado en una ventana emergente, dicha ventana debe mostrar la información en forma de pestañas.	

Tabla B.15. Historia de usuario # 16

Historia de usuario	
Número: 16	**Nombre:** Visualizar detalle de subcanal.
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1.5	**Iteración asignada:** 5
Programador responsable: Alik del Risco	
Descripción: El sistema debe permitir al usuario visualizar los detalles del subcanal seleccionado en una ventana emergente, dicha ventana debe mostrar la información en forma de pestañas.	

Tabla B.16. Historia de usuario # 17

Historia de usuario	
Número: 17	**Nombre:** Ejecutar en barrido.
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1.5	**Iteración asignada:** 6
Programador responsable: Alik del Risco	
Descripción: El sistema debe mostrar una ventana emergente que permita al usuario definir el tiempo que estará en barrido el dispositivo y el motivo de esta acción.	

Tabla B.17. Historia de usuario # 18

Historia de usuario	
Número: 18	**Nombre:** Ejecutar fuera de barrido.
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1.5	**Iteración asignada:** 6
Programador responsable: Alik del Risco	
Descripción: El sistema debe permitir al usuario ejecutar la acción de poner en fuera de barrido el dispositivo a través de una ventana emergente.	

Tabla B.18. Historia de usuario # 19

Historia de usuario	
Número: 19	**Nombre:** Notificar de forma sonora la existencia de alarma acorde con el SCADA SAINUX.
Usuario: Usuario	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.5	**Iteración asignada:** 7
Programador responsable: Alik del Risco	
Descripción: El sistema debe notificar de forma sonora al usuario la existencia de una alarma activa no reconocida en el SCADA SAINUX, el tono dependerá de la alarma con mayor criticidad.	

Tabla B.19. Historia de usuario # 20

Historia de usuario	
Número: 20	**Nombre:** Visualizar cada vista de sumario como máximo una sola vez.
Usuario: Usuario	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.5	**Iteración asignada:** 7
Programador responsable: Alik del Risco	

Continúa en la próxima página

Tabla B.19. Continuación de la página anterior

Descripción: El sistema debe asegurar la visualización una sola pestaña por cada tipo de sumario. Los tipos de sumario pueden ser de: alarmas, subcanales, dispositivos, puntos analógicos y digitales.

B.3. Diagrama de Secuencia

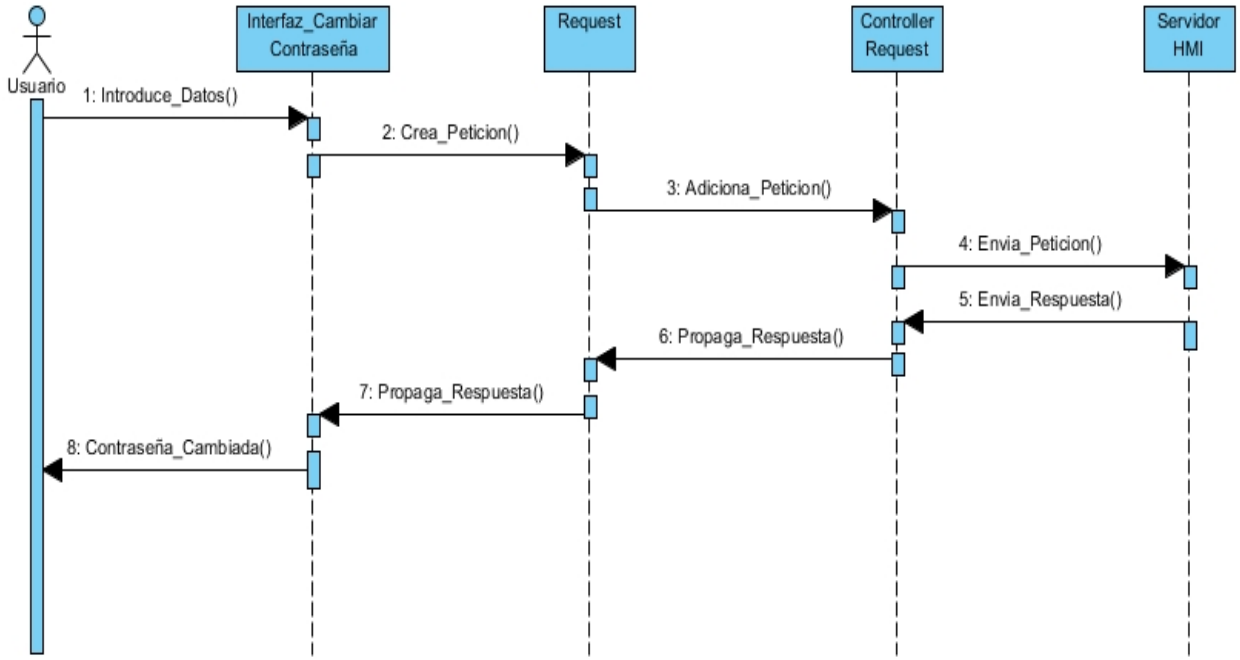

Figura B.5. Diagrama de Secuencia Cambiar Contraseña

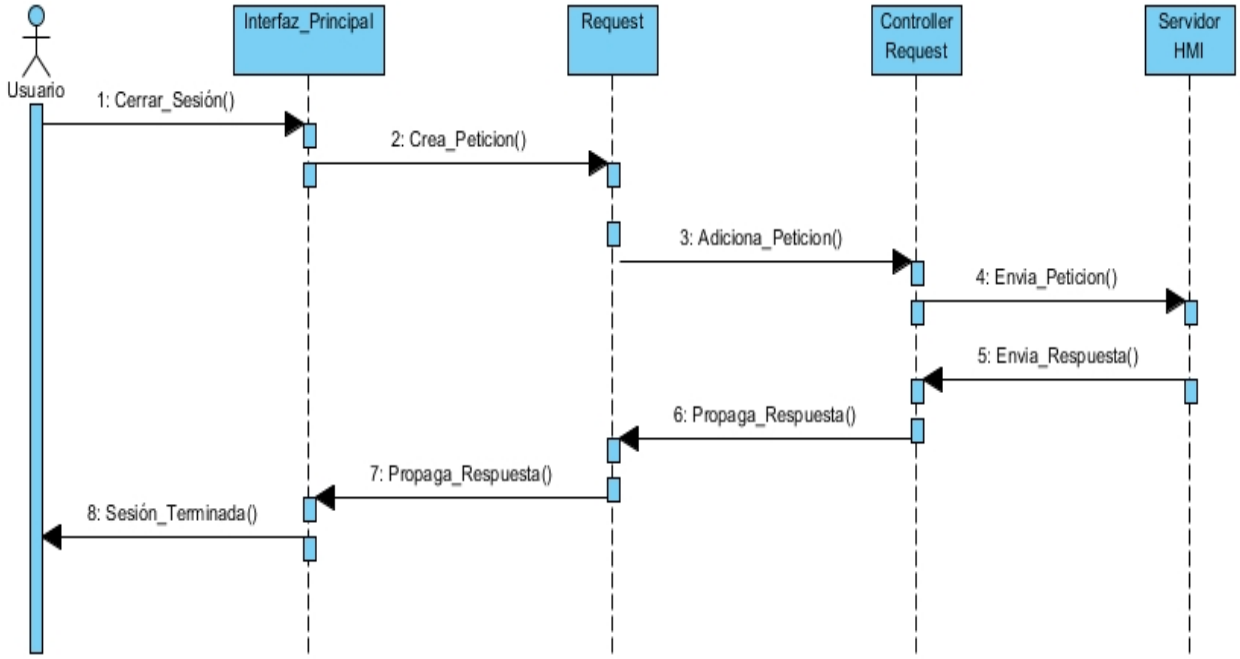

Figura B.6. Diagrama de Secuencia Cerrar Sesión

APÉNDICE C

Implementación y prueba

C.0.1. Pruebas de Aceptación.

Tabla C.1. Prueba de aceptación # 2

Caso de prueba de aceptación	
Código: HU2_P1	**Historia de usuario:** 2
Nombre: Visualizar datos de usuario.	
Descripción: El sistema debe visualizar los datos del usuario tales como el nombre de usuario y el tiempo restante de sesión.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución:	
Resultados esperados: En la barra de estado que se encuentra en la esquina derecha inferior se muestra los datos del usuario.	

Tabla C.2. Prueba de aceptación # 3

Caso de prueba de aceptación	
Código: HU3_P1	**Historia de usuario:** 3
Nombre: Autenticar usuario con perfiles.	
Descripción: El sistema debe mostrar una interfaz de autenticación para permitir al usuario introducir su usuario y contraseña, y poder solicitar los perfiles asociados.	
Condiciones de ejecución:	

Continúa en la próxima página

Tabla C.2. Continuación de la página anterior

Pasos de ejecución: El usuario introduce el usuario y la contraseña correctamente. El sistema realiza la petición con el servidor. El servidor envía la respuesta de la petición al sistema. El sistema muestra el listado de los perfiles asociados el usuario. El usuario selecciona el perfil y entra al sistema.
Resultados esperados: En la barra de estado que se encuentra en la esquina derecha inferior se muestra los datos del usuario.

Tabla C.3. Prueba de aceptación # 4

Caso de prueba de aceptación	
Código: HU4_P1	**Historia de usuario:** 4
Nombre: Cambiar contraseña.	
Descripción: El sistema muestra una interfaz que permite introducir la nueva contraseña.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: El usuario introduce el usuario, la contraseña actual y la nueva(dos veces) correctamente. El sistema realiza la petición con el servidor. El servidor envía la respuesta de la petición al sistema .	
Resultados esperados: Se muestra un mensaje indicando que el cambio de contraseña ha ocurrido correctamente.	

Tabla C.4. Prueba de aceptación # 5

Caso de prueba de aceptación	
Código: HU5_P1	**Historia de usuario:** 5
Nombre: Renovar sesión.	
Descripción: Cuando el tiempo de sesión del usuario autenticado en el sistema termina y este tiene permiso para renovar la sesión, el sistema le pregunta a través de una ventana emergente si desea renovar o cerrar la sesión.Solo se podrá renovar la sesión una sola vez.	
Condiciones de ejecución: El usuario debe estar autenticado. El usuario tiene permiso para renovar la sesión.	
Pasos de ejecución: El usuario acepta renovar la sesión. El sistema envía la petición al sevidor. El sistema renueva la sesión.	
Resultados esperados: Se muestra un mensaje indicando que el cambio de contraseña ha ocurrido correctamente.	

Tabla C.5. Prueba de aceptación # 6

Caso de prueba de aceptación	
Código: HU6_P1	**Historia de usuario:** 6
Nombre: Cerrar sesión.	
Descripción: El sistema permite cerrar la sesión.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: El usuario cierra la sesión. El sistema envía la petición al sevidor. El sistema cierra la sesión.	
Resultados esperados: Se cierra la sesión del usuario.	

Tabla C.6. Prueba de aceptación # 7

Caso de prueba de aceptación	
Código: HU7_P1	**Historia de usuario:** 7
Nombre: Visualizar sumario reducido de las alarmas.	
Descripción: El sistema debe mostrar siempre el sumario reducido de alarmas, mostrando las últimas 5 alarmas generadas por el sistema.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: Cada un segundo el sistema envía la petición al servidor. El sistema mediante el sumario reducido visualiza la respuesta de la petición enviada por el servidor.	
Resultados esperados: Los colores de las alarmas se muestran según el nivel de criticidad que tengan no se muestran correctamente, dicha no conformidad se corrige en la primera iteración destinada a la solución de dichos eventos. Logrando la correcta visualización de las ultimas 5 alarmas.	

Tabla C.7. Prueba de aceptación # 8

Caso de prueba de aceptación	
Código: HU8_P1	**Historia de usuario:** 8
Nombre: Visualizar sumario de alarmas.	
Descripción: El sistema debe permitir al usuario visualizar el estado de las alarmas, ordenas por criticidad del sistema.	
Condiciones de ejecución: El usuario debe estar autenticado.	

Continúa en la próxima página

Tabla C.7. Continuación de la página anterior

Pasos de ejecución: Cada un segundo el sistema envía la petición al servidor. El sistema mediante el sumario visualiza la respuesta de la petición enviada por el servidor.
Resultados esperados: Los colores de las alarmas se muestran según el nivel de criticidad que tengan no se muestran correctamente, además al mostrar el sumario mediante su correspondiente pestaña no se actualizaban los datos de las alarmas. Dichas no conformidades se corrigen en la primera iteración destinada a la solución de dichos eventos. Logrando la correcta visualización del sumario de alarmas.

Tabla C.8. Prueba de aceptación # 9

Caso de prueba de aceptación	
Código: HU9_P1	**Historia de usuario:** 9
Nombre: Visualizar sumario de puntos analógicos.	
Descripción: El sistema debe permitir al usuario visualizar el estado de los puntos analógicos.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: Cada un segundo el sistema envía la petición al servidor. El sistema mediante el sumario visualiza la respuesta de la petición enviada por el servidor.	
Resultados esperados: Al mostrar el sumario mediante su correspondiente pestaña no se actualizaban los datos de los puntos analógicos. Dicha no conformidad se corrige en la segunda iteración destinada a la solución de dichos eventos. Logrando la correcta visualización del sumario de puntos analógicos.	

Tabla C.9. Prueba de aceptación # 10

Caso de prueba de aceptación	
Código: HU10_P1	**Historia de usuario:** 10
Nombre: Visualizar sumario de puntos digitales.	
Descripción: El sistema debe permitir al usuario visualizar el estado de los puntos digitales.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: Cada un segundo el sistema envía la petición al servidor. El sistema mediante el sumario visualiza la respuesta de la petición enviada por el servidor.	

Continúa en la próxima página

Tabla C.9. Continuación de la página anterior

Resultados esperados: Al mostrar el sumario mediante su correspondiente pestaña no se actualizaban los datos de los puntos digitales. Dicha no conformidad se corrige en la segunda iteración destinada a la solución de dichos eventos. Logrando la correcta visualización del sumario de puntos digitales.

Tabla C.10. Prueba de aceptación # 11

Caso de prueba de aceptación	
Código: HU11_P1	**Historia de usuario:** 11
Nombre: Visualizar sumario de dispositivos.	
Descripción: El sistema debe permitir al usuario visualizar el estado de los dispositivos.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: Cada un segundo el sistema envía la petición al servidor. El sistema mediante el sumario visualiza la respuesta de la petición enviada por el servidor.	
Resultados esperados: Al mostrar el sumario mediante su correspondiente pestaña no se actualizaban los datos de los dispositivos. Dicha no conformidad se corrige en la tercera iteración destinada a la solución de dichos eventos. Logrando la correcta visualización del sumario de dispositivos.	

Tabla C.11. Prueba de aceptación # 12

Caso de prueba de aceptación	
Código: HU12_P1	**Historia de usuario:** 12
Nombre: Visualizar sumario de subcanales.	
Descripción: El sistema debe permitir al usuario visualizar el estado de los subcanales.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: Cada un segundo el sistema envía la petición al servidor. El sistema mediante el sumario visualiza la respuesta de la petición enviada por el servidor.	
Resultados esperados: Al mostrar el sumario mediante su correspondiente pestaña no se actualizaban los datos de los subcanales. Dicha no conformidad se corrige en la tercera iteración destinada a la solución de dichos eventos. Logrando la correcta visualización del sumario de subcanales.	

Tabla C.12. Prueba de aceptación # 13

Caso de prueba de aceptación	
Código: HU13_P1	**Historia de usuario:** 13
Nombre: Realizar paginado de sumarios.	
Descripción: El sistema debe permitir el paginado de los sumarios, de esta manera se dividen contenidos de gran longitud en varias páginas más cortas. La cantidad de datos a mostrar por página debe ser configurable.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: El usuario indica la cantidad de datos a mostrar por página.	
Resultados esperados: Se realiza el paginado de los sumarios.	

Tabla C.13. Prueba de aceptación # 14

Caso de prueba de aceptación	
Código: HU14_P1	**Historia de usuario:** 14
Nombre: Filtrar sumarios.	
Descripción: El sistema debe permitir filtrar los sumarios en dependencia de los campos que muestre cada sumario ó de algún valor en específico, de esta manera se pueden realizar búsquedas por categorías	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: El usuario indica mediante que valor va a realizar la búsqueda.	
Resultados esperados: Se realiza el filtrado de los sumarios.	

Tabla C.14. Prueba de aceptación # 15

Caso de prueba de aceptación	
Código: HU15_P1	**Historia de usuario:** 15
Nombre: Visualizar detalle de dispositivos.	
Descripción: El sistema debe permitir al usuario visualizar los detalles del dispositivo seleccionado.	
Condiciones de ejecución: El usuario debe estar autenticado. Seleccionar el dispositivo.	

Continúa en la próxima página

Tabla C.14. Continuación de la página anterior

Pasos de ejecución: El usuario selecciona el dispositivo que desee visualizar sus detalles. El sistema envía esta petición al servidor. El servidor envía al sistema los detalles del dispositivo.
Resultados esperados: Se visualiza los detalles del dispositivo.

Tabla C.15. Prueba de aceptación # 16

Caso de prueba de aceptación	
Código: HU16_P1	**Historia de usuario:** 16
Nombre: Visualizar detalle de subcanal.	
Descripción: El sistema debe permitir al usuario visualizar los detalles del subcanal seleccionado.	
Condiciones de ejecución: El usuario debe estar autenticado. Seleccionar el subcanal.	
Pasos de ejecución: El usuario selecciona el subcanal que desee visualizar sus detalles. El sistema envía esta petición al servidor. El servidor envía al sistema los detalles del subcanal.	
Resultados esperados: Se visualiza los detalles del subcanal.	

Tabla C.16. Prueba de aceptación # 17

Caso de prueba de aceptación	
Código: HU17_P1	**Historia de usuario:** 17
Nombre: Ejecutar en barrido.	
Descripción: El sistema debe permitir al usuario ejecutar la acción de poner en barrido el dispositivo.	
Condiciones de ejecución: El usuario debe estar autenticado. El dispositivo debe estar fuera de barrido.	
Pasos de ejecución: El usuario ejecuta la operación. El sistema envía esta petición al servidor. El servidor ejecuta la operación y envía los datos del estado del dispositivo.	
Resultados esperados: Se ejecuta la operación.	

Tabla C.17. Prueba de aceptación # 18

Caso de prueba de aceptación	
Código: HU18_P1	**Historia de usuario:** 18

Continúa en la próxima página

Tabla C.17. Continuación de la página anterior

Nombre: Ejecutar fuera de barrido.
Descripción: El sistema debe permitir al usuario ejecutar la acción de poner en fuera de barrido el dispositivo.
Condiciones de ejecución: El usuario debe estar autenticado. El dispositivo debe estar en barrido.
Pasos de ejecución: El usuario ejecuta la operación. El sistema envía esta petición al servidor. El servidor ejecuta la operación y envía los datos del estado del dispositivo.
Resultados esperados: Se ejecuta la operación.

Tabla C.18. Prueba de aceptación # 19

Caso de prueba de aceptación	
Código: HU19_P1	**Historia de usuario:** 19
Nombre: Notificar de forma sonora la existencia de alarma acorde con el SCADA SAINUX.	
Descripción: El sistema debe notificar de forma sonora al usuario la existencia de una alarma activa no reconocida en el SCADA SAINUX.	
Condiciones de ejecución: El usuario debe estar autenticado. Existencia de una alarma activa no reconocida.	
Pasos de ejecución:	
Resultados esperados: Sistema emite el sonido.	

Tabla C.19. Prueba de aceptación # 20

Caso de prueba de aceptación	
Código: HU20_P1	**Historia de usuario:** 20
Nombre: Visualizar cada vista de sumario como máximo una sola vez.	
Descripción: El sistema debe permitir al usuario visualizar cada vista de sumario como máximo una sola vez.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución:	
Resultados esperados: Sistema solomante visualiza un sumario como máximo de cada tipo.	